INDEX

COMPUTER SCIENCE (Code No. 083)

Learning Objectives:

☐ To understand basics of computers.

☐ To develop logic for Problem Solving.

☐ To develop problem solving skills and their implementation through using C++.

☐ To understand and implement the concept of Object Oriented Methodology.

☐ To understand the concept of working with Relational Database.

☐ To understand the basic concept of Computing Logic.

☐ To understand the basic concepts of Communication and Networking technologies.

☐ To understand Open Source Software.

Class XII (Theory) - C++

Duration: 3 hours **Total Marks: 70**

Unit No.	Unit Name	Marks
1.	OBJECT ORIENTED PROGRAMMING IN C++	30
2.	DATA STRUCTURE	14
3.	DATABASE MANAGEMENT SYSTEM AND SQL	8
4.	BOOLEAN ALGEBRA	8
5.	COMMUNICATION TECHNOLOGIES	10
Total		70

Unit 1: Object Oriented Programming in C++ (50 Theory + 40 Practical) Periods

REVIEW: C++ covered In Class - XI,

Object Oriented Programming: Concept of Object Oriented Programming - Data hiding, Data encapsulation, Class and Object, Abstract class and Concrete class, Polymorphism (Implementation of polymorphism using Function overloading as an example in C++); Inheritance, Advantages of Object Oriented Programming over earlier programming methodologies,

Implementation of Object Oriented Programming concepts in C++: Definition of a class, Member of a class - Data Members and Member Functions (methods), Using Private and Public visibility modes, default visibility mode (private); Member function definition: inside class definition and outside class definition using scope resolution operator (::); Declaration of objects as instances of a class; accessing members from object (s), Objects as function arguments-pass by value and pass by reference;

Constructor and Destructor: Constructor: special characteristics, declaration and definition of a constructor, default constructor, overloaded constructors, copy constructor, constructor with default arguments;

Destructor: Special Characteristics, declaration and definition of destructor;

Inheritance (Extending Classes): Concept of Inheritances, Base Class, Derived classes, protected visibility mode; Single level inheritance, Multilevel inheritance and Multiple inheritance, Privately

derived, publicly derived and Protectedly derived class, accessibility of members from objects and within derived class (es);

Data File Handling: Need for a data file, Types of data files - Text file and Binary file;

Text File: Basic file operations on text file: Creating/Writing text into file, Reading and Manipulation of text from an already existing text File (accessing sequentially).

Binary File: Creation of file, Writing data into file, Searching for required data from file, Appending data to a file, Insertion of data in sorted file, Deletion of data from file, Modification of data in a file;

Implementation of above mentioned data file handling in C++;

Components of C++ to be used with file handling:

Header file: fstream.h; ifstream, ofstream, classes;

Opening a text file in in, out, and app modes;

Using cascading operators (>><<) for writing text to the file and reading text from the file; open (), get (), read () put (), write(), getline() and close() functions; Detecting end-of-file (with or without using eof() function), tellg(), tellp(), seekg().seekp();

Pointers:

Introduction to Pointer, Declaration and Initialization of Pointer; Dynamic memory allocation/de-allocation operators: **new, delete;** Pointers and Arrays: Array of Pointers, Pointer to an array (1 dimensional array), Function returning a pointer, Reference variables and use of alias; Function call by reference. Pointer to structure: De-reference/Deference operator: *, ->; self referencial structure;

Unit 2: Data Structures (42 Theory + 36 Practical) Periods

Introduction to data structure- array, stack queues primitive and non-primitive data structure, linear and non-linear structure, static and dynamic data structure.

Arrays:

One and two Dimensional arrays: Sequential allocation and address calculation;

One dimensional array: Traversal, Searching (Linear, Binary Search), Insertion of an element in an array, deletion of an element from an array, Sorting (Insertion, Selection, Bubble)

Two-dimensional arrays: Traversal Finding sum/difference of two NxM arrays containing numeric values, Interchanging Row and Column elements in a two dimensional array;

Stack (Array and Linked implementation of Stack):

Introduction to stack (LIFO_Last in First out Operations)

Operations on stack (PUSH and POP) and its Implementation in C++, Converting expressions from INFIX to POSTFIX notation and evaluation of Postfix expression;

Queue: (Array and Linked Implementation)

Introduction to Queue (FIFO - First in First out operations)

Operations on Queue (Insert and Delete and its Implementation in C++, circular queue using array.

Unit 3: Database Management Systema and SQL

Common to both the options. Refer to unit 3 DATABASE AND SQL mentioned in case of Python for further details.

Unit 4: Boolean Algebra

Common to both the options. Refer to unit 4 mentioned in case of Python for further details.

Unit 5: Networking and Open Source Software

Common to both the options. Refer to unit 5 COMMUNICATION_TECHNOLOGIES mentioned in case of Python for further details.

Class XII (Practical) - C++

Duration: 3 hours **Total Marks : 30**

1. **Programming in C++** 12

   One programming problem in C++ to be developed and tested in Computer during the examination. Marks are allotted on the basis of following:

Logic	:	7 Marks
Documentation/Indentation	:	2 Marks
Output presentation	:	3 Marks

   Notes: The types of problem to be given will be of application type from the following topics

   - Arrays (One dimensional and two dimensional)
   - Class(es) and objects
   - Stack using arrays and or linked implementation
   - Queue using arrays (circular) and or linked implementation
   - Binary File operations (Creation, Displaying, Searching and modification)
   - Text File operations (Creation, Displaying and modification)

2. **SQL Commands** 05

   Five Query questions based on a particular Table / Relation to be tested practically on Computer during the examination. The command along with the result must be written in the answer sheet.

3. A digital circuit diagram (after reduction using k-map) to be given during the examination .The question must be written in the answer sheet. 02

4. **Project Work** 05

   The project has to be developed in C++ language with Object Oriented Technology and also should have use of Data files. (The project is required to be developed in a group of 2-4 students)

   - Presentation on the computer
   - Project report (Listing, Sample, Outputs, Documentations)
   - Viva

     * 1 mark is for innovation while writing programme.

5. **Practical File** 03+01*

Must have minimum 20 programs from the following topics

- Arrays (One dimensional and two dimensional, sorting, searching, merging, deletion' & insertion of elements)
- Class(es) and objects
- Stacks using arrays and linked implementation
- Queue using arrays & linked implementation (circular aslo).
- File (Binary and Text) operations (Creation, Updation, Query)
- Any computational Based problems
- 15 SQL commands along with the output based on any table/relation:

6. **Viva Voce** 02

Viva will be asked from syllabus covered in class XII and the project developed by student.

Guidelines for Projects (Class XI and XII)

1. **Preamble**

1.1 The academic course in Computer Science includes on Project in each year. The Purpose behind this is to consolidate the concepts and practices imparted during the course and to serve as a record of competence.

1.2 A group of 2-3 students as team may be allowed to work on one project.

2. **Project content**

2.1 Project for class XI can be selected from the topics mentioned in the syllabus or domains on the similar lines

2.2 Project for class XII should ensure the coverage of following areas of curriculum:

a) Flow of control

b) Data Structure

c) Object Oriented Programming C++

d) Data File Handling

Theme of the project can be

- Any subsystem of a System Software or Tool
- Any Scientific or a fairly complex algorithmic situation
- School Management, Banking, Library Information System, Hotel or Hospital Management System, Transport query system
- Quizzes / Games;
- Tutor, Computer Aided Learning Systems

2.3 It is suggested to prepare a bilingual (English and other Indian language) user manual part of project file.

2.4 The aim of the project is to highlight the abilities of algorithmic formulation, modular programming, optimized code preparation, systematic documentation and other associated aspects of Software Development.

Overview of C++

KEY POINTS:

Introduction to C++

- C++ is the successor of C language & developed by Bjarne Stroustrup at Bell Laboratories, New Jersey in 1979.

Tokens- smallest individual unit. Following are the tokens

- **Keyword**-Reserve word that have special meaning and can't be used as identifier
- **Identifiers-**Names given to different component of program such as variable, function, class, union etc.
- **Literals**-Value of specific data type that never change their value
- **Variable**- named storage location that can be manipulated during program run
- **Constant**- memory block where value can be stored once but can't changed later on
- **Operato**r – trigger some action on data
    - Arithmetic(+,-,*,/,%)
    - Relational/comparison (<,>,<=,>=,==,!=).
    - Logical(AND(&&),OR(||),NOT(!).
    - Conditional (? :)
    - Increment/Decrement Operators(++/--)
- **Precedence of operators:**

++(post increment),--(post decrement)	Highest ↓
++(pre increment),--(pre decrement),sizeof !(not),-(unary),+unary plus)	
*(multiply), / (divide), %(modulus)	
+(add),-(subtract)	
<(less than),<=(less than or equal),>(greater than), >=(greater than or equal to)	
==(equal),!=(not equal)	
&& (logical AND)	
||(logical OR)	
?:(conditional expression)	
=(simple assignment) and other assignment operators(arithmetic assignment operator)	
, Comma operator	Low

Data type- A specifier to create memory block of some specific size and type. For example – int,float,double,char etc.

cout – It is an object of **ostream_withassign** class defined in iostream.h header file and used to display value on monitor.

cin – It is an object of **istream_withassign** class defined in iostream.h header file and used to read value from keyboard for specific variable.

comment- Used for better understanding of program statements and escaped by the compiler to compile . *e.g.* – single line (//) and multi line(/*….*/)

Control structure :

Sequence control statement(if)	**conditional statement (if else)**	**case control statement (switch case)**	**loop control statement (while ,do… while, for)**
Syntax	Syntax	Syntax	Syntax

if(expression)
{
statements;
}
	If(expression)
	{
	statements;
	}
	else
	{
	statements;
	}
		switch(integral expression)
		{case (int const expr1):
		[statements
		break;]
		case (int const expr2):
		[statements,
		break;]
		default:Statements;}
			while(expression)
			{statements;}
			do ….while loop
			do
			{statement;}
			while(expression);
			for loop
			for(expression1;expression

			2;expression3) {statement;}

Note: any non-zero value of an expression is treated as true and exactly 0 (i.e. all bits contain 0) is treated as false.

Nested loop -loop within loop.

exit()- defined in process.h and used to leave from the program.

break- exit from the current loop.

continue- to skip the remaining statements of the current Iteration and passes control to the next iteration

goto- control is unconditionally transferred to the location of local label specified by <identifier>.

For example

A1:

cout<<"test";

goto A1;

Some Standard C++ libraries

Header File Name	**Purpose**	**Function**
iostream.h	Defines stream classes for input/output streams	get(), getline(), put(), eof()
stdio.h	Standard input and output	Gets(), puts(),getchar(),putchar()
ctype.h	Character tests	Isalpha(), islower(), toupper()
string.h	String operations	strcat(), strcmp(), strcpy(), strlen()
math.h	Mathematical functions	Sqrt(), pow(), exp(), sin(), floor()
stdlib.h	Utility functions	Abs(), malloc()

Some functions

- **isalpha(c)-**check whether the argument is alphabetic or not.
- **islower(c)-** check whether the argument is lowecase or not.
- **isupper(c)** - check whether the argument is uppercase or not.
- **isdigit(c)-** check whether the argument is digit or not.
- **isalnum(c)-** check whether the argument is alphanumeric or not.
- **tolower()-**converts argument in lowercase if its argument is a letter.
- **toupper(c)-** converts argument in uppercase if its argument is a letter.
- **strcat()-** concatenates two string.
- **strcmp-**compare two string if it is equal or not .
- **pow(x,y)-**return x raised to power y.
- **sqrt(x)-**return square root of x.
- **random(num)-**return a random number between 0 and (num-1)
- **randomize-** initializes the random number generator with a random value.

Array- Collection of variable of same data type that are referred by a common name.

One Dimension array

- An array is a continuous memory location holding variables of same data type in single row or single column

Two dimensional array

- A two diamensional array is a continuous memory location holding variables of same data type in of both row sand columns (like a matrix structure).

Function -Self-contained block of code that does some specific task and may return a value.

Function prototypes-Function declaration that specifies the function name, return type and parameter list of the function.

syntax: return_type function_name(type var1,type var2,….,type varn);

Passing value to function-

- Passing by value
- Passing by address/reference

Function overloading

- function name having several definition having parameter list

Function recursion

- Function that call itself either directly or indirectly.

Local variables

- Declared inside the function or a block.

Global variables

- Declared outside all braces { } in a program

Actual Parameters

parameter appear in function call statement used to send data from calling function to called function.

Formal Parameters

parameter appear in the function definition used to receive data from calling function.

Structure-Collection of logically related variable of different data types (Primitive and Derived) referenced under one name.

Nested structure

- A Structure definition within another structure.
- A structure containing object of another structure.

typedef

- Used to define new data type name
- typedef float amount ;

#**define Directives**

- Use to define a constant number or macro or to replace an instruction.
- #define Max 70
- #define SQUARE(a) a*a

Inline Function

- Inline functions are functions where the call statement is replace by actual code .
- **What happens when an inline function is written?**
- When the program is compiled, the code present in function body is replaced in the place of function call.

Solved questions

1 What is the difference between #define and const? Explain with suitable example.

2

Ans: While they both serve a similar purpose,#define and const act differently. When using #define the identifier gets replaced by the specified value by the compiler, before the code is turned into binary. This means that the compiler makes the substitution when you compile the application.

Eg:

#define number 100

In this case every instance of “number” will be replaced by the actual number 100 in your code, and this means the final compiled program will have the number 100 (in binary).

#define with different types of data:

- The #define preprocessor allows us to define symbolic names and constants.

Eg: #define PI 3.14159

- The #define allows you to make textsubstitutions before compiling the program.

Eg: #define MAX 70
Before compilation, if the C++ preprocessor finds MAX as one word, in the source code, it replaces it with the number 70.

- The #define preprocessor can be used in the creation of macros (code substitution).

Eg:
#define SQUARE(x) x*x
Before compilation, if the C++ preprocessor finds SQUARE(x), where x is any value in the source code, it replaces it with its square (ie x*x). Here a macro substitutes text only; It does not check for data types. On the other hand, when we use const and the application runs, memory is allocated for the constant and the value gets replaced when theapplication is run.
Syntax: const type variable_name=value;
Eg:
const int a=10;
The value of a constant is fixed and in the above example, the value for a in entire program is 10 only. You cannot change the value of a, since it is declared as constant.

2. What is the purpose of using a typedef command in C++? Explain with suitable example. 2

Ans: C++ allows you to define explicitly new data type names by using the keyword typedef. Using typedef does not actually create a new data class, rather it defines a new name for an existing type. This can increase the portability of a program as only the typedef statements would have to be changed. Typedef makes your code easier to read and understand. Using typedef can also aid in self documenting your code by allowing descriptive names for the standard data types. The syntax of the typedef statement is typedef type name; Where type is any C++ data type and name is the new name for this type. This defines another name for the standard type of C++.
For example, you could create a new name for float values by using the following statement: typedef float amount;

3. Illustrate the use of #define in C++ to define a macro.
Ans: The #define preprocessor can be used in the creation of macros (code substitution).
Eg:
#define SQUARE(x) x*x
Before compilation, if the C++ preprocessor finds SQUARE(x),where x is any value in the source code, it replaces it with its square (ie x*x). Here a macro substitutes text only; It does not check for data types.

2.a) Define the term Data Encapsulation in the context of Object Oriented Programming.Give a suitable example using a C++ code to illustrate the same. 2

Ans: Encapsulation is wrapping up of characteristics and behavior into one unit.
A class binds together data and its associated functions under one unit .
Eg:

```
class Rectangle
{ private: float len,bre,area;
public: void readData( )
{ cout<<"\nEnter the length and
breadth..";
cin>>len>>bre;
}
};
```

Eg:
Here in the above class the data members ie len,bre,area and the member functions ie readData() are bind together in a class named as Rectangle. Ie The member functions can access any data member in the class.

Benefits with encapsulation:
(i) Modularity.
(ii) Information hiding.

1 Marks questions

1) Name the header files that shall be needed for the following code:

```
void main( )
{ char word[]="Board Exam";
  cout<<setw(20)<<word;
}
```

2) Name the Header file to which the following built in functions belongs to:
(i) gets() (ii) abs() (iii) sqrt() (iv) open()

2 Marks questions:

1) **Rewrite the following program after removing the syntactical error(s) if any.Underline each correction.** **CBSE 2012**

```
#include<iostream.h>
Class Item
{
long IId, Qty;
public:
 void Purchase { cin>>IId>>Qty;}
  void Sale()
   {
     cout<<setw(5)<<IId<<"Old:"<< Qty<<endl;
     cout<< "New :"<<Qty<<endl;
   }};
void main()
{
Item I;
Purchase();
I.Sale()
}
```

2) Find the output of the following program:

```
#include<iostream.h>
#include<ctype.h>
typedef char Str80[80];
void main()
{char *Notes;
 Str80 str= " vR2GooD";
int L=6;
Notes =Str;
```

```
while(L>=3)
 {
 Str[L]=(isupper(Str[L])? tolower(Str[L]) : toupper(Str[L]));
 cout<<Notes<<endl;
 L--;
 Notes++;
 }}
```

3) Observe the following program and find out, which output(s) out id (i) to (iv) will not be expected from program? What will be the minimum and maximum value assigned to the variables Chance?

```
#include<iostream.h>
#include<stdlib.h>
void main()
{
 randomize();
 int Arr[] = {9,6}, N;
 int Chance = random(2)+10;
 for(int c=0;c<2;c++)
 {
        N= random(2);
        cout<<Arr[N]<<"#";
 }}
```

i) 9#6#
ii) 19#17#
iii) 19#16#
iv) 20#16#

3 Marks questions:

4) Find the output of the following program:

```
#include<iostream.h>
class METRO
{
int Mno, TripNo, PassengerCount;
public:
METRO(int Tmno=1) { Mno =Tmno; PassengerCount=0;TripNo=0}
void Trip(int PC=20) { TripNo++, PassengerCount+=PC};

void StatusShow()
 {
cout<<Mno<< ":"<<TripNo<< " :"<<PassengerCount<<endl;}
}};
void main()
{
METRO M(5), T;
M.Trip();
M.StatusShow();
```

```
T.Trip(50);
T.StatusShow();
M.Trip(30);
M.StatusShow();
}
```

Ans : 5: 1: 20
1: 1: 50
5: 2: 50

5) **Rewrite the following program after removing the syntactical error(s) if any.Underline each correction.**

```
#include<iostream.h>
void main( )
{  F = 10, S = 20;
   test(F;S);
   test(S);
}
 void test(int x, int y = 20)
{ x=x+y;
count<<x>>y;
}
```

6) **Rewrite the following program after removing syntactical error(s) if any. Underline each correction.**

```
#include “iostream.h”
Class MEMBER
{ int Mno;
float Fees;
PUBLIC:
void Register ( )  {cin>>Mno>>Fees;}
void Display( )  {cout<<Mno<<" : "<<Fees<<endl;}
};
void main()
{ MEMBER delete;
Register();
delete.Display();
}
```

7) Find the output for the following program:

```
#include<iostream.h>
#include<ctype.h>
void encrypt ( char T[ ])
{ for( int i=0 ; T[i] != ‘ \0’ ; i += 2)
   if( T[i] = = ‘A’ || T[i] = = ‘E’ )
    T[i] = ‘#’ ;
  else if (islower (T[i] ))
      T[i] = toupper(T[i]);
  else
    T[i] = ‘@’;}

void main()
{ char text [ ] = “SaVE EArTh in 2012”;
```

```
 encrypt(text);
  cout<<text<<endl;
}
```

8) Find the output of the following program:

```
#include<iostream.h>
void main( )
{ int U=10,V=20;
for(int I=1;I<=2;I++)
{ cout<<"[1]"<<U++<<"&"<<V- 5 <<endl;
 cout<<"[2]"<<++V<<"&"<<U + 2 <<endl; } }
```

Function Overloading

- A function name having several definitions that are differentiable by the number or types of their arguments is known as **function overloading.**

Example : A same function **print()** is being used to print different data types:

```
#include <iostream.h>
class printData
{
  public:
    void print(int i) {  cout << "Printing int: " << i << endl;      }
    void print(double  f) {  cout << "Printing float: " << f << endl;  }
    void print(char *c) {  cout << "Printing character: " << c << endl;  }
};
int main(void)
{
  printData pd;
  pd.print(5);    // Call print to print integer
  pd.print(500.263);  // Call print to print float
  pd.print("Hello C++");   // Call print to print character
  return 0;
}
```

When the above code is compiled and executed, it produces following result:

```
Printing int: 5
Printing float: 500.263
Printing character: Hello C++
```

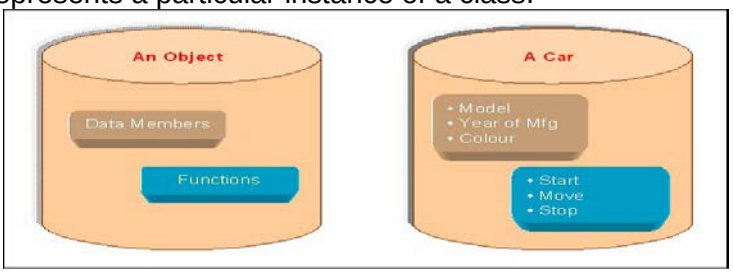
BASIC CONCEPTS OF OOP & CLASSES AND OBJECTS

Object-Oriented Programming groups related data and functions together in a class, generally making data private and only some functions public. Restricting access decreases coupling and increases cohesion. It has proven to be very effective in reducing the complexity increase with large programs. For small programs may be difficult to see the advantage of OOP over, eg, structured programming because there is little complexity regardless of how it's written.

It helps in programming approach in order to built robust user friendly and efficient software and provides the efficient way to maintain real world software. OOP is an Object Oriented Programming language which is the extension of Procedure Oriented Programming language. OOP reduce the code of the program because of the extensive feature of Inheritence and Polymorphism. OOP have many properties such as Data Hiding, Inheritance Data Abstraction, Data Encapsulation and many more. The main aim is to write program to solve a problem by defining and using different object play role in system.

Objects:

Object is the basic unit of object-oriented programming. Objects are identified by its unique name. An object represents a particular instance of a class.



An Object is a collection of data members and associated member functions also known as methods.

Classes:

- Classes are blueprint or data types which defines characteristics and behaviou of same type of objects.
- Thus a Class represents a set of individual objects. Characteristics of an object represented in a class are also known as Properties. The actions that can be performed by objects become functions of the class and are referred to as Methods.
- Classes are the blueprints upon which objects are created. E.g when we design a map of the house, the architect first designs it. Once it is designed, the raw material is used to build the house. In this example, Design of the house is CLASS (blueprint) and the house built on the basis of design is an object.

No memory is allocated when a class is created. Memory is allocated only when an object is created, i.e., when an instance of a class is created. for example:

```
 class student
{
int enroll;
char name[20];
char CLASS[7];
public:
void input();
};
 student s1;
```

here student is a class and s1 is an object.

Inheritance:

- Inheritance is the process of forming a new class from an existing class or base class. The base class is also known as parent class or super class. The new class that is formed is called derived class.
- Derived class is also known as a child class or sub class. Inheritance helps in reducing the overall code size of the program, which is an important concept in object-oriented programming. It is the process by which one class inherits the properties of another class.

Data Abstraction:

- Data Abstraction represents only the needed feature without presenting the background details. Abstraction is achieved by using encapsulation and data hiding.

The concept of abstraction relates to the idea of hiding data that are not needed for presentation.
for example:
Break system in a car provide paddle to the user for operation and hide the internal mechanical system that can be either hydrolic or wired system.

An **Abstract Data Type** is defined as a data type that is defined in terms of the operations that it supports and not in terms of its structure or implementation.

Encapsulation:

- Encapsulation combines data and functions into a single unit called class.

Data hiding is a process of hiding some properties and methods from outside the class and it is implemented by using private and protected visibility mode.

```
class MyClass
{ public:
int sample();
int example(char *se)
int endfunc();
..... //Other member functions
private:
int x;
float sq;
..... //Other data members
};
```

In the above example, the data members integer x, float sq are private data members and member functions sample(),example(char* se),endfunc() are public members .

Polymorphism:

Poly means many and **morphs** mean form, so polymorphism means one name multiple form. There are two types of polymorphism: compile time and run time polymorphism. An operator or function can perform operation differently with different situation. Polymorphism in c++ is implemented by using function overloading, function overriding or virtual function.

Overloading:

- Overloading is one type of Polymorphism.
- It allows an operator/ function to have different meanings, depending on its context.

- When an exiting operator or function begins to operate on new data type, or class, it is understood to be overloaded.

.

OOP Term	Definition
Method	Same as function, but the typical OO notation is used for the call, i.e. f(x,y) is written x.f(y) where x is an object of class that contains this f method.
Send a message	Call a function (method)
Instantiate	Allocate a class/struct object (ie, instance) with new
Class	A struct with both data and functions
Object	Memory allocated to a class/struct. Often allocated with new.
Member	A field or function is a member of a class if it's defined in that class
Constructor	A member function having same name as that of the class that initializes data members of an object at the time of creation of the object.
Destructor	Function-like code that is called when an object is deleted to free any resources (e.g. memory) that is has pointers to. It is automatically invoked when object goes out of scope.
Inheritance	Deriving properties of parent class to the child class.
Polymorphism	Defining functions with the same name, but different parameters.
Overload	A function is overloaded if there is more than one definition. See polymorphism.
Sub class	Same as child, derived, or inherited class.
Super class	Same as parent or base class.
Attribute	Same as data member or member field

IMPLEMENTATION OF OOPS IN C++

The major components of Object Oriented Programming are . **Classes & Objects**

A **Class i**s a group of similar objects . **Objects** share two things that is properties and *behavior*. For example : Dogs have properties (name, color, breed, hungry) and behavior (barking,
fetching, wagging tail). Identifying the properties and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming. These real-world observations are translate into the world of object-oriented programming.

Software objects are conceptually similar to real-world objects: they too consist of properties and related behavior. An object stores its properties in *fields* (variables in some programming languages) and exposes its behavior through functions.

classes in Programming :

- It is a collection of variables, often of different types and its associated functions
- class defines a blueprint for a data type.

Declaration/Definition :

A class definition starts with the keyword **class** followed by the class name; and the class body,
enclosed by a pair of curly braces. A class definition must be followed either by a semicolon or a
list of declarations.

```
class  class_name {
                        access_specifier_1:
                        member1;
                        access_specifier_2:
                        member2;
                        ...
                        } object_names;
```

Where class_name is a valid identifier for the class, object_names is an optional list of names for objects of this class. The body of the declaration can contain members that can either be data or function declarations, and optionally access specifiers.

[Note: the default access specifier is private.

Example : class Box

```
{ int a;
public:
   double length;   // Length of a box
   double breadth;  // Breadth of a box
   double height;   // Height of a box
};
```

Access specifiers

Access specifiers are used to identify access rights for the data and member functions of the class. There are three main types of access specifiers in C++ programming language:

- private
- public
- protected

Member-Access Control

Member-Access Control

Type of Access	Meaning
Private	Class members declared as **private** can be used only by member functions and friends (classes or functions) of the class.
Protected	Class members declared as **protected** can be used by member functions and friends (classes or functions) of the class. Additionally, they can be used by classes derived from the class.
Public	Class members declared as **public** can be used by any function.

Importance of Access Specifiers

Access control helps prevent you from using objects in ways they were not intended to be used. Thus it helps in implementing data hiding and data abstraction.

OBJECTS in C++:

Objects represent instances of a class. Objects are basic run time entities in an object oriented system.

Creating object / defining the object of a class:

The general syntax of defining the object of a class is:-

Class_name object_name;

In C++, a class variable is known as an object. The declaration of an object is similar to that of a variable of any data type. The members of a class are accessed or referenced using object of a class.

```
Box Box1;        // Declare Box1 of type Box
Box Box2;        // Declare Box2 of type Box
```

Both of the objects Box1 and Box2 will have their own copy of data members.

Accessing / calling members of a class

All member of a class are private by default.

Private member can be accessed only by the function of the class itself. Public member of a class can be accessed through any object of the class. They are accessed or called using object of that class with the help of dot operator (.).

Object_name.Data_member=value;

The general syntax for accessing member function of a class is:-

Object_name. Function_name (actual arguments);

The dot ('. ') used above is called the **dot operator or class member access operator**. The dot
operator is used to connect the object and the member function. The private data of a class can be accessed only through the member function of that class.

Class methods definitions (Defining the member functions)

Member functions can be defined in two places:-

- **Outside the class definition**

The member functions of a class can be defined outside the class definitions. It is only declared
inside the class but defined outside the class. The general form of member function definition
outside the class definition is:

```
 Return_type Class_name:: function_name (argument list)
{
  // Function body
}
```

Where symbol :: is a scope resolution operator.

The scope resolution operator (::) specifies the class to which the member being declared
belongs, granting exactly the same scope properties as if this function definition was directly
included within the class definition

The member function of a class can be declared and defined inside the class definition.

```
class sum
{
int A, B, Total;
public:
void getdata ()
{
cout< ”\n enter the value of A and B”;
cin>>A>>B;
}
void display ();
};
void sum::display ()
{
total = A+B;
cout<<”\n the sum of A and B=”<<total;
}
```

Differences between struct and classes in C++

In C++, a *structure* is a class defined with the **struct** keyword. Its members are public by default and default derivation mode is public. A class defined with the **class** keyword. Its members are private by default and default derivation mode is private . This is the only difference between structs and classes in C++.

INLINE FUNCTIONS

inline functions definition starts with keyword inline
The compiler replaces the function call statement with the function code itself(expansion) and then compiles the entire code.
They run little faster than normal functions as function calling overheads are saved.
A function can be declared inline by placing the keyword inline before it.

4 Marks Questions

Q 1**) Define a class TAXPAYER in C++ with following description :**

Private members :

- Name of type string
- PanNo of type string
- Taxabincm (Taxable income) of type float
- TotTax of type double
- A function CompTax( ) to calculate tax according to the following slab:

Taxable Income	Tax %
Upto 160000	0
>160000 and <=300000	5
>300000 and<=500000	10
>500000	15

Public members :

A parameterized constructor to initialize all the members
A function INTAX( ) to enter data for the tax payer and call function CompTax( ) to assign TotTax.
A function OUTAX( ) to allow user to view the content of all the data members.

Ans.

```
class TAXPAYER
{
char Name[30],PanNo[30];
float Taxabincm;
double TotTax;
void CompTax()
{ if(Taxabincm >500000)
TotTax= Taxabincm*0.15;
else if(Taxabincm>300000)
TotTax= Taxabincm*0.1;
Else if(Taxabincm>160000)
TotTax= Taxabincm*0.05;
else
```

```
TotTax=0.0; }
public:
TAXPAYER(char nm[], char pan[], float tax, double ttax) //parameterized constructor
{ strcpy(Name,nm);
strcpy(PanNo,pan);
Taxabincm=tax;
TotTax=ttax; }
void INTAX()
{ gets(Name);
cin>>PanNo>>Taxabincm;
CompTax(); }
void OUTAX()
{ cout<<Name<<'\n'<<PanNo<<'\n'<<Taxabincm<<'\n'<<TotTax<<endl; }
};
```

Q2.) **Define a class HOTEL in C++ with the following description:**

Private Members

Rno	//Data member to store room number
Name	//Name of customer
Tariff	//To store per day charges
NOD	//number of days
CALC	//A function to calculate and return amount as NOD*Tariff and if value of NOD*Tariff >10000 then as 1.05*NOD*Tariff

Public Members:

Checkin( )	//A function to enter the content Rno,Name,Tariff and NOD
Checkout()	// A function to display all data members and calls CALC function

Solution

```
#include<iostream.h>
class HOTEL
{
unsigned int Rno;
char Name[25];
unsigned int Tariff;
unsigned int NOD;
int CALC()
{       int x;
        x=NOD*Tariff;
        if( x>10000)
        return(1.05*NOD*Tariff);
        else
        return(NOD*Tariff);
}
public:
void  Checkin()
{cin>>Rno>>Name>>Tariff>>NOD;}
```

```
        void Checkout()
        {cout<<Rno<<Name<<Tariff<<NOD<<CALC();}
        };
```

Q. 3 Define a class Applicant in C++ with following description:
Private Members

- A data member ANo (Admission Number) of type long
- A data member Name of type string
- A data member Agg(Aggregate Marks) of type float
- A data member Grade of type char
- A member function GradeMe( ) to find the Grade as per the Aggregate Marks obtained by a student. Equivalent Aggregate marks range and the respective Grades are shown as follows

Aggregate Marks	Grade
> = 80	A
Less than 80 and > = 65	B
Less than 65 and > = 50	C
Less than 50 and > = 33	D
Less than 33	E

Public Members

- A function Enter( ) to allow user to enter values for ANo, Name, Agg & call function GradeMe( ) to find the Grade
- A function Result ( ) to allow user to view the content of all the data members.

Ans.

```
class Applicant
{       long  ANo;
        char Name[25];
        float Agg;
        char Grade;
        void GradeMe( )
        {       if (Agg > = 80)
                        Grade = 'A';
                else if (Agg >= 65 && Agg < 80 )
                        Grade = 'B';
                else if (Agg >= 50 && Agg < 65 )
                        Grade = 'C;
                else if(Agg>=33 && Agg <50)
                        Grade = 'D';
                else
                        Grade= 'E';
        }
public:
        void Enter ( )
        {       cout <<"\n  Enter Admission No.       ";   cin>>ANo;
                cout <<"\n  Enter Name of  the Applicant      ";  cin.getline(Name,25);
                cout <<"\n  Enter Aggregate Marks obtained by the Candidate :";  cin>>Agg;
                GradeMe( );
        }

        void Result( )
        {       cout <<"\n  Admission No.     "<<ANo;
                cout <<"\n  Name of  the Applicant    ";<<Name;
                cout<<"\n  Aggregate Marks obtained by the Candidate.     " << Agg;
```

```
                cout<<\n    Grade Obtained  is    " << Grade ;
        }
};
```

Q4. **Define a class Sports in C++ with following descriptions:**
Private members:
S_Code of type long
S_Name of type character array (String)
Fees of type integer
Duration of type integer

Public members:
Constructor to assign initial values of S_Code as 1001, S_Name as "Cricket", Fees as 500, Duration 70, A function NewSports() which allows user to enter S_Code, S_Name and Duration. Also assign the values to Fees as per the following conditions:

S_Name	Fees
Table Tennis	2000
Swimming	4000
Football	3000

A function DisplaySports() to display all the details.

Constructors and Destructors

- Constructors and destructors are special member functions of classes that are used to construct and destroy class objects. Construction may involve memory allocation and initialization for objects.
- Destruction may involve cleanup and deallocation of memory for objects.

The following restrictions apply to constructors and destructors:

- Constructors and destructors do not have return type, not even void
- They can not return values.
- References and pointers cannot be used on constructors and destructors because their addresses cannot be taken.
- Constructors cannot be declared with the keyword virtual.
- Constructors and destructors cannot be declared static, const, or volatile.
- Unions cannot contain class objects that have constructors or destructors.
- The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope.
- A constructor does not allocate memory for the class object its **this** pointer refers to, but may allocate storage for more objects than its class object refers to. If memory allocation is required for objects, constructors can explicitly call the new operator. During cleanup, a destructor may release objects allocated by the corresponding constructor. To release objects, use the delete operator.
- Derived classes do not inherit constructors or destructors from their base classes, but they do call the constructor and destructor of base classes.
- Constructors are also called when local or temporary class objects are created, and destructors are called when local or temporary objects go out of scope.
- We can call member functions from constructors or destructors.
- Constructor is a member function with the same name as its class.
  For example:

```
class X {
        public:
                X();      // constructor for class X
```

```
};
```

- Destructors are usually used to deallocate memory and do other cleanup for a class object and its class members when the object is destroyed.
- A destructor is called for a class object when that object passes out of scope or is explicitly deleted.
- A destructor is a member function with the same name as its class prefixed by a ~ (tilde).

  For example:

```
class X { public:
            X();      // Constructor for class X
           ~X();      // Destructor for class X
          };
```

Default Constructors and Destructors

If you don't declare a constructor or a destructor, the compiler makes one for you. The default constructor and destructor take no arguments and do nothing.

What good is a constructor that does nothing? In part, it is a matter of form. All objects must be constructed and destructed, and these do-nothing functions are called at the right time.

Using constructors and destructors.

```
// Demonstrates declaration of a constructors and
// destructor for the Cat class
#include <iostream.h>          // for cout
class Cat                      // begin declaration of the class
{  public:                                  // begin public section
   Cat(int initialAge);        // constructor
   ~Cat();                     // destructor
   int GetAge();               // accessor function
   void SetAge(int age);       // accessor function
   void Meow();
 private:                      // begin private section
   int itsAge;                 // member variable
};
Cat::Cat(int initialAge)          // constructor definition  of Cat,
{    itsAge = initialAge; }
Cat::~Cat()  { cout<<"destructor is called"; }                  // destroy the object of cat
when it is no longer referred.
int Cat::GetAge()
{    return itsAge;}
void Cat::SetAge(int age)
{ itsAge = age;}  //      set member variable its age to value passed in by parameter
age}
void Cat::Meow()
{    cout << "Meow.\n";}
int main()
{   Cat Frisky(5);
   Frisky.Meow();
   cout << "Frisky is a cat who is " ;
   cout << Frisky.GetAge() << " years old.\n";
   Frisky.Meow();
   Frisky.SetAge(7);
   cout << "Now Frisky is " ;
   cout << Frisky.GetAge() << " years old.\n";
   return 0;
}
```

Output:
Meow.
Frisky is a cat who is 5 years old.
Meow.
Now Frisky is 7 years old.
destructor is called

Copy Constructor

- A copy constructor is a special constructor in the C++ programming language used to create a new object as a copy of an existing object.
- Normally the compiler automatically creates a copy constructor for each class (known as a default copy constructor) but for special cases the programmer creates the copy constructor, known as a user-defined copy constructor. In such cases, the compiler does not create one.
- Copying of objects is achieved by the use of a copy constructor and a assignment operator. A copy constructor has as its first parameter a reference to its own class type. It can have more arguments, but the rest must have default values associated with them. The following would be valid copy constructors for class X:

```
X(const X& copyFromMe);
X(X& copyFromMe);
X(const X& copyFromMe, int = 10);
X(const X& copyFromMe, double = 1.0, int = 40);
```

The following cases may result in a call to a copy constructor:

- When an object is returned by value
- When an object is passed (to a function) by value as an argument
- When an object is returned from the function
- When an object is caught
- When an object is placed in a brace-enclosed initializer list

An object can be assigned value using one of the two techniques:

- Explicit assignment in an expression
- Initialization

Explicit assignment in an expression

```
Object A;
Object B;
A = B;      // translates as Object::operator=(const Object&), thus A.operator=(B) is called
            // (invoke simple copy, not copy constructor!)
```

Initialization

An object can be initialized by any one of the following ways.

a. **Through declaration**

Object B = A; // translates as Object::Object(const Object&) (invoke copy constructor)

b. Through function arguments

type function (Object a);

c. Through function return value

Object a = function();

The copy constructor is used only for initializations, and does not apply to assignments where the assignment operator is used.

The implicit copy constructor of a class calls base copy constructors and copies its members by means appropriate to their type. If it is a class type, the copy constructor is called. By using a user-defined copy constructor the programmer can define the behavior to be performed when an object is copied.

Examples

These examples illustrate how copy constructors work and why they are required sometimes.

Implicit copy constructor

Let us consider the following example.

```
//copy constructor
#include <iostream>
class Person
        {       public:
                int age;
                Person(int a) {age=a;}
        };
 int main()
        {Person timmy(10);
        Person sally(15);
        Person timmy_clone = timmy;
        cout << timmy.age << " " << sally.age << " " << timmy_clone.age << endl;
        timmy.age = 23;
        cout << timmy.age << " " << sally.age << " " << timmy_clone.age << endl;
        return 0; }
       Output
       10 15 10
        23 15 10
```

As expected, *timmy* has been copied to the new object, *timmy_clone*. While *timmy's* age was changed, *timmy_clone's* age remained the same. This is because they are totally different objects.

The compiler has generated a copy constructor for us, and it could be written like this:

```
Person( Person& copy)
  { age=copy.age; }
```

INHERITANCE

- Inheritance is the process by which new classes (*derived* classes) are created from existing classes (*base* classes).
- The derived classes have all the features of the base class and the programmer can add new features specific to the newly created derived class.

Features or Advantages of Inheritance:

Reusability:

- Inheritance helps the code to be reused in many situations.
- The base class is defined and once it is compiled, it need not be reworked.
- Using the concept of inheritance, the programmer can create as many derived classes from the base class as needed while adding specific features to each derived class as needed**.**

Saves Time and Effort:

The above concept of reusability achieved by inheritance saves the programmer time and effort. The main code written can be reused in various situations as needed.

Increases Program Structure which results in greater reliability.

General Format for implementing the concept of Inheritance:

class derived_classname: access specifier baseclassname

For example, if the *base* class is *MyClass* and the derived class is sample it is specified as:

```
class sample: public MyClass
```

The above makes sample have access to both *public* and *protected* variables of base class *MyClass*.

Reminder about public, private and protected access specifiers:

1. If a member or variables defined in a class is private, then they are accessible by members of the same class only and cannot be accessed from outside the class.
2. Public members and variables are accessible from outside the class.
3. Protected access specifier is a stage between private and public. If a member functions or variables defined in a class are protected, then they cannot be accessed from outside the class but can be accessed from the derived class.
   Inheritance Example:

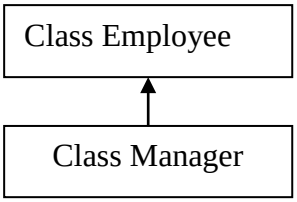
```
class MyClass
{ int x;
 public:
   MyClass(void) { x=0; }
   void f(int n1)
  {  x= n1*5;}
 void output(void) { cout<<x; }
    };
class sample: public MyClass
{
int s1;
 public:
   sample(void) { s1=0; }
 void f1(int n1)
 { s1=n1*10;}
 void output(void)
 { MyClass::output();
  cout << "\t"<<s1; }
 };
 int main(void)
 {sample s;
 s.f(10);
 s.output();
 s.f1(20);
 s.output();
 }
```

The output of the above program is
50 0
50 200

Types of Inheritance

1. Single class Inheritance:

- Single inheritance is the one where you have a single base class and a single derived class.



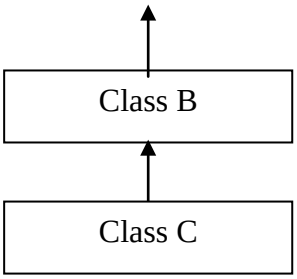
it is a Base class (super)

it is a sub class (derived)

2. Multilevel Inheritance:

- In Multi level inheritance, a class inherits its properties from another derived class.

it is a Base class (super) of B

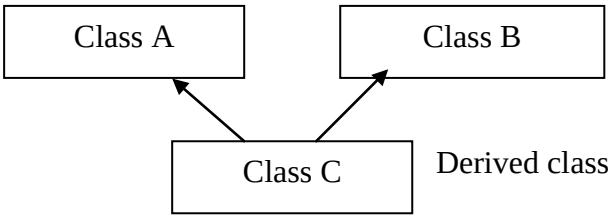


it is a sub class (derived) of A and base class of class C

derived class(sub) of class B

3. Multiple Inheritances:

- In Multiple inheritances, a derived class inherits from multiple base classes. It has properties of both the base classes.

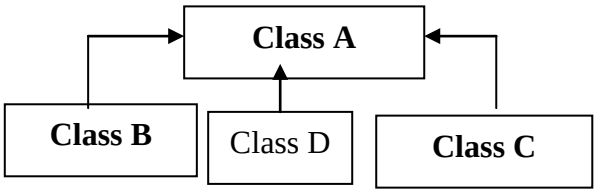


Base class

Derived class

4. Hierarchical Inheritance:

- In hierarchial Inheritance, it's like an inverted tree. So multiple classes inherit from a single base class. It's quite analogous to the File system in a unix based system.

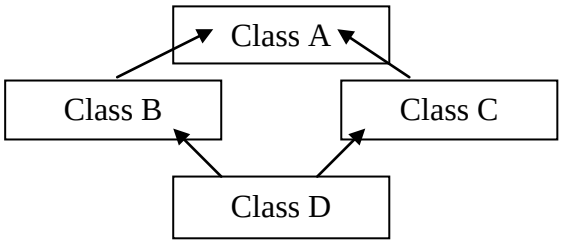


Base class (super)

Sub Class(derived)

5. Hybrid Inheritance:

- In this type of inheritance, we can have mixture of number of inheritances but this can generate an error of using same name function from no of classes, which will bother the compiler to how to use the functions.
- Therefore, it will generate errors in the program. This has known as ambiguity or duplicity.
- Ambiguity problem can be solved by using **virtual base classes**



2 Marks Question

Practice 1 :- Answer the questions after going through the following class.

```
class Exam
{ char Subject[20] ;
        int Marks ;
public :
        Exam()                                  // Function 1
```

```
    {strcpy(Subject, "Computer" ) ; Marks = 0 ;}
    Exam(char P[ ])                          // Function 2
    {strcpy(Subject, P) ;
    Marks=0  ;
    }
   Exam(int M)                               // Function 3
  {strcpy(Subject, "Computer") ; Marks = M ;}
   Exam(char P[ ], int M)                    // Function 4
  {strcpy(Subject, P) ; Marks = M ;}
  };
```

a) Which feature of the Object Oriented Programming is demonstrated using Function 1, Function2, Function 3 and Function 4 in the above class Exam?

Ans:- Function Overloading (Constructor overloading)

b) Write statements in C++ that would execute Function 3 and Function 4 of class Exam.

Ans:- Exam a(10); and Exam b("Comp", 10);

Practice 2: Consider the following declaration :

```
class welcome
{ public:
        welcome (int x, char ch);      // constructor with parameter
        welcome();                     // constructor without parameter
        void  compute();
private:
        int x;    char ch;
};
```

which of the following are valid statements

```
welcome obj (33, 'a');
welcome obj1(50, '9');
welcome obj3(8);
obj1= welcome (45, 'T');
obj3= welcome;
```

Ans. Valid and invalid statements are

welcome obj (33, 'a');	**valid**
welcome obj1(50, '9');	**valid**
welcome obj3(8);	**invalid**
obj1= welcome (45, 'T');	**valid**
obj3= welcome;	**invalid**

		Inheritance Mode		
		public	protected	private
Members in Base Class	public	public	protected	private
	protected	protected	protected	private
	private	X	X	X
		Members in derived class		

Access	public	protected	Private
members of the same class	yes	Yes	Yes
members of derived classes	yes	Yes	No
By objects	yes	No	No

Practice 3: What do you mean by visibility modes? What is their effect on inheritance?

Practice 4: Explain different types of inheritance.

4 Marks Questions

Practice1 :- Consider the following declarations and answer the questions given below:

```
class vehicle
        { int wheels;
        protected:
        int passenger;
        public:
        void inputdata( int, int);
        void outputdata();};
        class heavyvehicle: protected vehicle
        {int dieselpetrol;
        protected:
        int load;
        public:
        void readdata( int, int);
        void writedata();};
        class bus:private heavyvehicle
        {char marks[20];
        public:
        void fetchdata(char);
        void displaydata();};
```

(i) Name the base class and derived class of the class **heavyvehicle.**
(ii) Name the data members that can be accessed from function **displaydata()**
(iii) Name the data members that can be accessed by an object of **bus class**
(iv) Is the member function outputdata() accessible to the objects of **heavyvehicle class.**

Ans

(i) Base class = vehicle and derived class = bus
(ii) The data members passenger, load, marks are available to function display data
(iii) No data members can be accessed by the object of bus calss.
(iv) No member functions outputdata () is not accessible to the objects of heavy vehicle class.

Practice2 :- Consider the following declarations and answer the questions given below:

```
#include <iostream.h>
        class book
        {char title[20];
        char author[20];
        int noof pages;
        public:
                void read();
                void show();};
        class textbook: private textbook
        {int noofchapters, noofassignments;
        protected:
        int standard;
        public:
```

```
void readtextbook();
void showtextbook();};
class physicsbook: public textbook
{char topic[20];
public:
void readphysicsbook();
void showphysicsbook();};
```

(i) Name the members, which can be accessed from the member functions of class physicsbook.
(ii) Name the members, which can be accessed by an object of Class textbook.
(iii) Name the members, which can be accessed by an object of Class physicsbook.
(iv) What will be the size of an object (in bytes) of class physicsbook.

Ans

(i) standard , readtextbook(),showtextbook(),readphysicsbook(),showphysicsbook() and topic;
(ii) readtextbook() and showtextbook()
(iii) readphysicsbook(), showphysicsbook(), readtextbook() and showtextbook()
(iv) The size of object of physicsbook= size of book + size of Textbook + size of physicsbook.
= 42+6+20 = 68 bytes

Practice3 : Answer the questions (i) to (iv) based on the following:

```
Class CUSTOMER
{       int Cust_no;
        char Cust_Name[20];
        protected:
        void Register();
public:
        CUSTOMER( );
        void Status( );};
class SALESMAN
{       int Salesman_no;
        char Salesman_Name[20];
        protected:
          float Salary;
        public:
        SALESMAN( );
        void Enter( );
        void Show( );};
class SHOP : private CUSTOMER, public SALESMAN
{               char Voucher_No[10];
                char Sales_Date[8;
        public :
                SHOP( );
                void Sales_Entry( );
                void Sales_Detail( );};
```

(i) Write the names of data members, which are accessible from object belonging to class CUSTOMER.
(ii) Write the names of all the member functions which are accessible from object belonging to class SALESMAN.
(iii) Write the names of all the members which are accessible from member functions of class SHOP.

(iv) How many bytes will be required by an object belonging to class SHOP?

DATA FILE HANDLING IN C++

Key Points:

- Text file: A text file stores information in ASCII characters. In Text file some internal character translation takes place while writing or reading the file, such as end of line ('\n') is converted into carriage and return characters.
- Binary file: A binary file contains information in the non-readable form i.e. in the same format in which it is stored in memory.
- Stream: A stream is a general term used to name flow of data. Different streams are used to represent different kinds of data flow.
- There are three file I/O classes used for file read / write operations.
  - **ifstream** - can be used for read operations.
  - **ofstream** - can be used for write operations.
  - **fstream** - can be used for both read & write operations.
- **fstream.h**:
- This header file includes the definitions for the stream classes ifstream, ofstream and fstream. In C++ **file input output** facilities implemented through fstream.h header file.
- It contain predefines set of operation for handling file related input and output, fstream class ties a file to the program for input and output operation.
- A file can be opened using:
- **By the constructor method**. This method is preferred when a first file is opened by the object at the time of creation of object.
- Example :
- **ofstream file("student.dat"); or ofstream file("student.dat",ios::out); for text file**
- **ofstream file("student.dat",ios::out | ios::binary); for binary file**
-
- **ifstream file("student.dat"); or ifstream file("student.dat",ios::in); for text file**
- **fstream file("student.dat",ios::in | ios::binary); for binary file**
- Note that if file opening mode is not supplied as parameter the constructor takes ios::out as its default opening mode for the object of ofstream class and ios:: in is used for object of ifstream class. In case of object of fstream we must pass file opening mode in the constructor as it takes no default file opening mode.
- **By the open() function of the stream.** It will be preferred when another file is required to be open after closing the previous opened file by the same object. Although we can use open function to open first file also by the object.

  Example:

  fstream file;

  file.open("book.dat", ios::in | ios::out | ios::binary);

- **File modes:**
  - **ios::out** It creates file in output mode and allows writing into the file.
  - **ios::in** It creates file in input mode and permit reading from the file.
  - **ios::app** To retain the previous contents of the file and to append to the end of existing file.
  - **ios::ate** To place the file pointer at the end of the file, but you can write data any where in the file.
  - **ios::trunc** It truncates the existing file (empties the file).
  - **ios::nocreate** If file does not exist this file mode ensures that no file is created and open() fails.
  - **ios::noreplace** If file does not exist, a new file gets created but if the file already exists, the open() fails.
  - **ios::binary –** Opens a file in binary mode.

eof(): This function determines the end-of-file by returning true(non-zero) for end of file otherwise returning false(zero).

open():If you want to manage multiple file with same stream use open().

Stream_object.open("Filename",(Filemode));

e.g., fstream fio;

fio.open("book.dat", ios::in | ios::out | ios::binary);

The operator **|** is known as bitwise-OR operator.

Text File functions:

Character I/O:

get() – There is two overloaded get() function.

```
char ch;
file.get(ch);          to read one character from file
char s[80];
file.get(s, 80);       to read a line of text or up to 80 character
```

put() - writing a single character in textfile e.g. **file.put(ch);**

getline() - read a line of text from text file store in a buffer.

e.g

```
char s[80];
file.getline(s, 80);
```

The difference between get(s, 80) and getline(s,80) is that in get() function new line character is not extracted from stream where as in getline() function new line is automatically extracted from stream.

We can also use **file>>ch** for reading and **file<<ch** writing in text file. But >> operator does not accept white spaces.

close(): This function terminates the connection between the file and stream associated with it.

Stream_object.close();

Binary file functions:

read(): The read() function reads a fixed number of bytes from the specified stream and puts them in the buffer.

Stream_object.read((char *)& Object, sizeof(Object));

write(): The write() function writes fixed number of bytes from a specific memory location to the specified stream.

Stream_object.write((char *)& Object, sizeof(Object));

Note:
Both functions take two arguments.
- The first is the address of variable, and the second is the length of that variable in bytes. The address of variable must be type cast to type char*(pointer to character type)
- The data written to a file using write( ) can only be read accurately using read( ).

file pointer: the file pointer indicates the position in the file at which the next input/output is to occur.

Functions used for moving file pointer

seekg(): It places the file pointer to the specified position in a stream before input operation.

seekp(): It places the file pointer to the specified position in a stream before output operation.

tellg(): This function returns the current position of the file pointer in a stream.

tellp(): This function returns the current position of the file pointer in a stream.

Steps To Create A File

- Declare an object of the desired file stream class(ifstream, ofstream, or fstream)
- Open the required file to be processed using constructor or open function.
- Process the file.
- Close the file stream using the object of file stream.

General program structure used for creating a Text File

To create a text file using strings I/O

```
#include<fstream.h>    //header file for file operations
void main()
{
char s[80], ch;
ofstream file("myfile.txt");     //open myfile.txt in default output mode
if(!file)
    cout<<"myfile.txt can not be open";
else
  {
  do
            {
            cout<<"\n enter line of text";
            gets(s); //standard input
            file<<s;// write in a file myfile.txt
            cout<<"\n more input y/n";
            cin>>ch;
            }while(ch!='n'||ch!='N');
  file.close();
  }
} //end of main
```

To create a text file using characters I/O

```
#include<fstream.h>    //header file for file operations
void main()
{
char  ch;
ofstream file("myfile.txt");     //open myfile.txt in default output mode
```

```
if(!file)
    cout<<"myfile.txt can not be open";
else
  {
   do{
            ch=getche();
            if (ch==13)  //check if character is enter key
                    cout<<'\n';
            else
            file<<ch;  // write a character in text file 'myfile.txt '
      } while(ch!=27); // check for escape key
  file.close();
  }
 }//end of main
```

Text files in input mode:

To read content of 'myfile.txt' and display it on monitor.

```
#include<fstream.h>    //header file for file operations
void main()
{
char  ch;
ifstream file("myfile.txt");     //open myfile.txt in default input mode
if(!file)
   cout<<"myfile.txt can not be open";
else
   {while(file)
            {
            file.get(ch)  // read a character from text file 'myfile.txt'
            cout<<ch;  // write a character in text file 'myfile.txt '
            }
  file.close();
  }
} //end of main
```

Exercise: <u>1 Mark Questions</u>

1. **Observe the program segment carefully and answer the question that follows:**

```
class item
{int item_no;
char item_name[20];
public:
void enterDetail( );
void showDetail( );
int getItem_no( ){ return item_no;}
};
void modify(item x, int y )
{fstream File;
File.open( "item.dat", ios::binary | ios::in | ios::out) ;
item i;
int recordsRead = 0, found = 0;
while(!found && File.read((char*) &i , sizeof (i)))
        {recordsRead++;
```

```
            if(i. getItem_no( ) == y )
                {_____//Missing statement
                File.write((char*) &x , sizeof (x));  found = 1;
                }
            }
        if(! found)
        cout<<"Record for modification does not exist" ;
        File.close() ;
        }
```

If the function modify( ) is supposed to modify a record in the file " item.dat ", which item_no is y, with the values of item x passed as argument, write the appropriate statement for the missing statement using seekp( ) or seekg( ), whichever is needed, in the above code that would write the modified record at its proper place.

If the function modify( ) modifies a record in the file " item.dat " with the values of item x passed as argument, write the appropriate parameter for the missing parameter in the above code, so as to modify record at its proper place.

Ans.1. File.seekp((recordsRead-1)*sizeof(x),ios::beg); or
File.seekp(-1*sizeof(x),ios::cur);

General program structure used for operating a Text File

2 Marks Questions

Text files in input mode:

Write a function in a C++ to count the number of lowercase alphabets present in a text file "BOOK.txt".

```
int countalpha()
{       ifstream Fin("BOOK.txt");
        char ch;
        int count=0;
        while(!Fin.eof())
                {Fin.get(ch);
                if (islower(ch))
                        count++;
                }
        Fin.close();
        return count;
}
```

Function to calculate the average word size of a text file.

```
    void calculate()
    {   fstream File;
        File.open("book.txt",ios::in);
        char a[20];
        char ch;
        int i=0,sum=0,n=0;
        while(File)
        {   File.get(ch);
            a[i]=ch;
            i++;
            if((ch==' ') || ch(== '.')||(char==',')(ch=='\t')||(ch=='\n')
                {i --;  sum=sum +i;
                i=0;  N++;
            }
```

```
        }
    cout<<"average word size is "<<(sum/n);
    }
```

Assume a text file "coordinate.txt" is already created. Using this file create a C++ function to count the number of words having first character capital.

```
int countword()
{       ifstream Fin("BOOK.txt");
        char ch[25];
        int count=0;
        while(!Fin.eof())
                {Fin>>ch;
                if (isupper(ch[0]))
                        count++;
                }
        Fin.close();
        return count;
}
```

Function to count number of lines from a text files (a line can have maximum 70 characters or ends at '.')

```
int countword()
{       ifstream Fin("BOOK.txt");
        char ch[70];
        int count=0;
        if (!Fin)
        {       cout<<"Error opening file!" ;
                exit(0);
        }
        while(1)
        {Fin.getline(ch,70,'.');
                if (Fin.eof())
                        break;
                count++;
        }
        Fin.close();
        return count;
}
```

A program to display the size of a file in bytes.

```
#include<iostream.h>
#include<fstream.h>
#include<process.h>
#include<conio.h>
int main()
{
        char filename[13];
        clrscr();
        cout<"Enter Filename:\n";
        cin.getline(filename,13);
        ifstream infile(filename);
        if(!infile)
                {cout>>"sorry ! Can not open "<<filename <<"file\n";
                exit(-1);
                }
```

```
        long no_bytes=0;
        char ch;
        infile.seekg(0,ios::end);
        no_bytes=infile.tellg();
        cout<<"File Size is"<<no_bytes<<"bytes\n";
        return 0;
}
```

Text files in output mode:

C++ program, which initializes a string variable to the content "There is an island of opportunity in the middle of every difficulty." and output the string one character at a time to the disk file "OUT.TXT".

```
#include<fstream.h>
int main()
{       ofstream fout("OUT.TXT");
        char *str = "There is an island of opportunity in the middle of every difficulty." ;
        int i=0;
        if(!fout)
        {
                cout<<"File cannot be opened ";
                return 0;
        }
        while (str[i]!='\0')
        {fout<<str[i];
                i++;
        }
        fout.close();
}
```

Exercise: (2 Marks Questions)

1. te a function in a C++ to count the number of uppercase alphabets present in a text file "BOOK.txt"
2. Write a function in a C++ to count the number of alphabets present in a text file "BOOK.txt"
3. Write a function in a C++ to count the number of digits present in a text file "BOOK.txt"
4. Write a function in a C++ to count the number of white spaces present in a text file "BOOK.txt"
5. Write a function in a C++ to count the number of vowels present in a text file "BOOK.txt"
6. Write a function in a C++ to count the average word size in a text file "BOOK.txt"
7. Write a function in C++ to print the count of the word "the" as an independent word in a text file STORY.TXT.
   For example, if the content of the file STORY.TXT is
   There was a monkey in the zoo.
   The monkey was very naughty.
   Then the output of the program should be 2.
8. Assume a text file "Test.txt" is already created. Using this file, write a function to create three files "LOWER.TXT" which contains all the lowercase vowels and "UPPER.TXT" which contains all the uppercase vowels and "DIGIT.TXT" which contains all digits.
9. Create a function FileLowerShow() in c++ which take file name(text files)as a argument and display its all data into lower case
10. Write a function in C++ to count the number of lines present in a text file "Story.txt".

HOTS FILE HANDLING

1. Write a function in a C++ to count the number of consonants present in a text file "Try.txt"
2. Write a function in a C++ to count the number of uppercase vowels present in a text file "Novel.txt"
3. Write a function in a C++ to display the sum of digits present in a text file "Fees.txt".
4. Write a function in a C++ to display the product of digits present in a text file "Number.txt".
5. Write a function in a C++ to find the largest digit present in a text file "Marks.txt"

3 Marks Questions

General program structure used for operating a Binary File

Program to read and write a structure using read() and write() using binary file.

```
struct student
{
char name[15];
float percent;
};
void main()
{
        clrscr();
        student s;
        strcpy(s.name,"rasha");
        s.percent=89.50;
        ofstream fout;
        fout.open("saving", ios::out | ios:: binary);
        if(!fout)
        {
                cout<<"File can't be opened";
                break;
        }
        fout.write((char *) &s, sizeof(student));
        fout.close();
        ifstream fin;
        fin.open("saving",ios::in | ios:: binary);
        fin.read((char *) & s,sizeof(student));
        cout<<s.name;
        cout<<"\n has the percent: "<<s.percent;
        fin.close();
}
```

Function to add more objects belonging to class JOKE at the end of JOKES.DAT file.

```
class JOKE{int jokeid; char type[5], jokedesc[200];
                public:
                void Newjokeentry(){cin>>jokeid>>type; cin.getline(jokedesc,200);}
                void showjoke(){cout<<jokeid<<"\t"<<type<<endl<<jokedesc<<endl;}
                };
void append()
{
        fstream afile;
        afile.open("JOKES.DAT", ios::binary | ios::app);
```

```
        JOKE j;
        int n,i;
        cout<<"How many objects you want to add :";
        cin>>n;
        for (i=0;i<n;i++)
        {
                j.Newjokeentry();
                afile.write((char *)& j, sizeof (JOKE));
        }
        afile.close();
}
```

Given a binary file TELEPHONE.DAT, containing records of the following class Directory

```
class Directory
{       char name[20],address[30], areacode[5], phone_no[15];
public:
        void register();
        void show();
        int checkcode(char AC[ ]) { return strcmp(areacode, AC);}
};
```

Write a function COPYABC() in C++, that would copy all those records having areacode as "123" from TELEPHONE.DAT to TELEBACK.DAT

```
COPYABC()
{ifstream ifile("TELEPHONE.DAT",ios::in|ios::binary);
 If(!ifle) { cout<<"could not open TELEPHONE.DAT"; exit(-1);}
else
        {ofstream ofile("TELEBACK",ios::out|ios::bainary);
        if(!ofile)  {cout<<"could not open TELEBACK.DAT"; exit(-1);}
        else
                {Directory d;
                while(ifile.read((char *)&d, sizeof(d)))
                        {if(d.checkcode("123")==0)
                                Ofile.write((char *)&d,sizeof(d));
                        }
                ifile.close();
                ofile.close();
                }
        }
}
```

Exercise :3 Marks Question

2. **Write a function in C++ to search for a BookNo from a binary file "BOOK.DAT", assuming the binary file is containing the objects of the following class.**

```
        class BOOK
        {
        int Bno;
        char Title[20];
        public:
        int RBno(){return Bno;}
        void Enter(){cin>>Bno;gets(Title);}
        void Display(){cout<<Bno<<Title<<endl;}
        };
```

3. **Write a function in C++ to add new objects at the bottom of a binary file "STUDENT.DAT", assuming the binary file is containing the objects of the following class.**

```
class STUD
{int Rno;
char Name[20];
public:
void Enter()
{cin>>Rno;gets(Name);}
void Display(){cout<<Rno<<Name<<endl;}
};
```

POINTERS

Pointer is a memory variable which stores address of an object of specified data type.
It supports dynamic allocation routines.
It can improve the efficiency of certain routines.

C++ Memory Map

Code Segment: It holds the executable instructions of the exe program.
Data Segment: It holds the Global variables(variables which remain in the memory as long as program continues.)
Stack Segment: It is used for holding return addresses at function calls, arguments passed to the functions, local variables for functions. It also stores the current state of the CPU.
Heap/Free Store : It is a region of free memory from which chunks of memory are allocated via dynamic memory allocation during program execution(runtime).

Static Memory Allocation:

The amount of memory to be allocated is known in advance and it allocated during compilation, it is referred to as Static Memory Allocation. For example

```
int a=5;        //2 bytes is allocated  to variable a at compile time
int r[10];      //20 contiguous bytes is allocated  to array r at compile time
```

Dynamic Memory Allocation:

The amount of memory required to be allocated during program execution(runtime) is referred as dynamic memory allocation.
There are two operators available in C++ for Dynamic Memory Allocation **new** and **delete**

```
int *x=new int; //new operator will allocate 2 bytes in heap and assign address of the allocated memory to pointer x
delete x;               //de-allocate the memory allocated at address stored in x
int *p=new int[10];   //new operator will allocate 20 contiguous bytes(array of 10 integers) in heap and //assign address of the first byte of the allocated memory to pointer p
delete [ ]p;            // de-allocates the memory allocated(20 bytes here) at address stored in p
```

Declaration and initialization of pointer variables:

```
Syntax :    Data_type  *pointer_name;
e.g.        int *x;
```

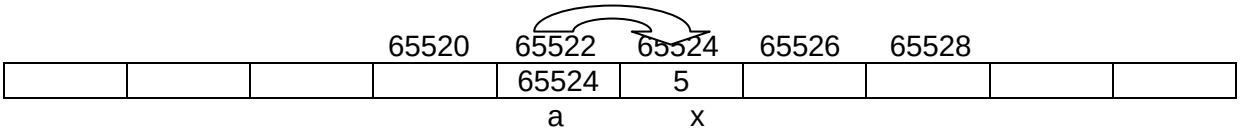
For example:

```
#include<iostream.h>
void main()
{int x=5;
int *a;//here 'a' is a pointer to int which can store address of an object of type int
a=&x;//address of x is assigned to pointer 'a'
cout<<"Value of x is : "<<x<<endl;
cout<<"Address of x is : "<<&x<<endl;
cout<<"Address stored in a is : "<<a<<endl;
cout<<"Values stored at memory location pointed by a is : "<<*a<<endl;
cout<<"Address of a is :"<<&a<<endl;
}
```

output:

```
Value of x is : 5
Address of x is : 65524
Address stored in a is : 65524
Value stored at memory location pointed by a is : 5
```

Address of a is : 65522
In the above example the **&x** gives address of x which in this case happen to be 65524 and **dereference operator *** gives the value stored in the memory location stored in pointer variable 'a' as shown in the figure below.



Pointers and Arrays in C++

The concept of array is very much bound to the one of pointer. In fact, the identifier of an array is equivalent to the address of its first element, as a pointer is equivalent to the address of the first element that it points to, so in fact they are the same concept. For example, supposing these two declarations:

```
int b[20];
int * p;
```

The following assignment operation would be valid:

```
p = b;
```

After that, p and b would be equivalent and would have the same properties. The only difference is that we could change the value of pointer p by another one, whereas b will always point to the first of the 20 elements of type int with which it was defined. Therefore, unlike p, which is an ordinary pointer, b is an array, and an array can be considered a constant pointer. Therefore, the following assignment statement would not be valid:

```
b= p;
```

Because b is an array, so it operates as a constant pointer, and we cannot assign values to constants. Due to the characteristics of variables, all expressions that include pointers in the following example are perfectly valid:

```
#include <iostream.h>
int main ()
{int b[5];
int *p;
p = b;
p[0] = 10; // p[0] and *p refers to the same value i.e. b[0]
p++;
*p = 20;
p = &b[2];
*p = 30;
p = b + 3;
*p = 40;
p = b;
*(p+4) = 50;     //*(p+4) and p[4] refers to the same value i.e. b[4]
p=b;
for (int n=0; n<5; n++)
        cout << p[n] << ", ";
cout<<endl;
for (int n=0; n<5; n++)
        cout << *(b+n) << ", ";
return 0;
}
```

Output is

```
10, 20, 30, 40, 50,
10, 20, 30, 40, 50,
```

Note: A pointer variable can be used as an array as in above example both *p and p[0] refers to the same variable similarly b[0] and *b refers to the same variable. Again p[1] and *(p+1) are same.

Address calculation method is same for both pointer and array. For example

```
int a[5]={2,4,6,7,9};
int *q=a;
```

The address of a[index] is calculated as

Base address of a (i.e. address of a[0]) + index * sizeof (data type of a)

for example if the base address of a is 1000 then address of a[3] is calculated as

&a[3] =1000 + 3 * sizeof (int) = 1000 + 3 * 2=1000 + 6 = 1006

Note that a[3] and *&a[3] both will give the same value i.e. 7

Similarly address of (q+3) is calculated as

Address stored in q + 3 * sizeof (data type of pointer belongs to in this case int)

(q+3) = 1000 + 3 * sizeof (int) = 1000 + 3 * 2=1006

Arithmetic operation on pointers

We can increment or decrement a pointer variable for example

```
float a[5]={3.0,5.6,6.0,2.5,5.3};
float *p=a;
++p;
--p;
```

if the base address of a is 1000 then statement ++p will increment the address stored in pointer variable by 4 because p is a pointer to float and size of a float object is 4 byte, since ++p is equivalent to p=p+1 and address of p+1 is calculated as

Address stored in p + 1 * sizeof (float)= 1000 + 1 * 4 = 1004.

We can add or subtract any integer number to a pointer variable because adding an integer value to an address makes another address e.g.

```
void main()
{int p[10]={8,6,4,2,1};
Int *q;
q=p;//address of p[0] is assigned to q assuming p[0] is allocated at memory location 1000
q=q+4;//now q contains address of p[4] i.e. 1000+4*sizeof(int)=1000+4*2= 1008
cout<<*q<<endl;
q=q-2;//now q contains address of p[2] i.e. 1008-2*sizeof (int)=1008-2*2=1004
cout<<*q<<endl;
 }
```

Output is

```
1
4
```

Addition of two pointer variables is meaningless.

Subtraction of two pointers is meaningful only if both pointers contain the address of different elements of the same array

For example

```
void main()
{int p[10]={1,3,5,6,8,9};
int *a=p+1,*b=p+4;
p=p+q;      //error because sum of two addresses yields an illegal address
int x=b-a;      //valid
cout<<x;
}
```

Output is

```
3
```

In the above example if base address of a is 1000 then address of a would be 1002 and address of b would be 1008 then value of b-a is calculated as

(Address stored in b-address stored in a)/sizeof(data type in this case int)

(1008-1002)/2=3

Multiplication and division operation on a pointer variable is not allowed
We cannot assign any integer value other than 0 to a pointer variable.
For example
```
int *p;
p=5;    //not allowed
p=0;    //allowed because 0 is a value which can be assigned to a variable of any data type
```

Null Pointer
A null pointer is a regular pointer of any pointer type which has a special value that indicates that it is not pointing to any valid reference or memory address. This value is the result of type-casting the integer value zero to any pointer type.
```
int *q=0;   // q has a NULL pointer value
float * p;
p=NULL;    // p has a NULL (NULL is defined as a constant whose value is 0) pointer value
```

Note: 0 memory address is a memory location which is not allocated to any variable of the program.

void pointers
void pointer is a special pointer which can store address of an object of any data type. Since void pointer do not contain the data type information of the object it is pointing to we can not apply dereference operator * to a void pointer without using explicit type casting.
```
void main()
{
int x=5;
float y=3.5;
int *p;
float *q;
void *r;
p=&y;  //error cannot convert float * to int *
q=&x;  //error cannot convert int * to float *
p=&x;  //valid
cout<<*p <<endl;  //valid
q=&y;  //valid
cout<<*q<<endl ;       //valid
r=&x;  //valid
cout<<*r<<endl;  //error pointer to cannot be dereference without explicit type cast
cout<<*(int *)r<<endl; // valid and display the values pointed by r as int
r=&y;   //valid
cout<<*(float *)r<<endl; //valid and display the values pointed by r as float
}
```

Note: Pointer to one data type cannot be converted to pointer to another data type except pointer to void

Array of pointers and pointer to array
An array of pointer is simply an array whose all elements are pointers to the same data type.
For example
```
Void main()
{
float x=3.5,y=7.2,z=9.7;
float *b[5]; // here b is an array of 5 pointers to float
b[0]=&x;
```

```
b[1]=&y;
b[2]=&z;
cout<<*b[0]<<"\t"<<*b[1]<<"\t"<<*b[2]<<endl;
cout<<sizeof(b)<<sizeof(b[0])<<sizeof(*b[0]) ;
 }
```

Output is

```
3.5       7.2       9.7
10        2          4
```

In the above example the expression sizeof(b) returns the number of bytes allocated to b i.e. 10 because array b contain 5 pointers to float and size of each pointer is 2 bytes. The expression sizeof(b[0]) returns the size of first pointer of the array b i.e. 2bytes. The expression sizeof(*b[0]) returns the size of the data type the pointer b[0] points to which is float, so the expression returns 4 as output.

Pointer to array is a pointer which points to an array of for example

```
void main()
{
float a[10], b[5],c[10];
float  (*q)[10]; //here q is a pointer to array of 10 floats
q=&a; //valid
q=&c;  //valid
q=&b;  //error cannot convert float [5]* to float [10]*
cout<<sizeof(q)<<"\t"sizeof(*q)
}
```

After removing the incorrect statement q=&b from the program the output of the program is

```
2      40
```

Because q is a pointer and pointer to any thing is and address which is of 2bytes. But sizeof(*q) returns the size of an array of 10 floats i.e. 40 because *q points to an array of 10 floats.

Pointers to pointers

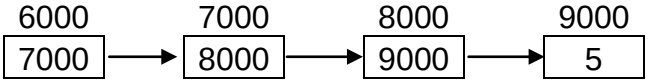
C++ allows the use of pointers that point to pointers, that these, in its turn, point to data (or even to other pointers). In order to do that, we only need to add an asterisk (*) for each level of reference in their declarations:

```
Void main()
{
int x=5;
int *a;  // a is pointer to int
int **b;  //b is pointer to pointer to int
int ***c;  //c is pointer to pointer to pointer to int
a=&x;
b=&a;
c=&b;
cout<<x<<"\t"<<*a<<"\t"<<**b<<"\t"<<***c<<endl;
}
```

Output is

```
5        5        5        5
```

Assuming variable x, a, b, c are allocated randomly at memory location 9000, 8000, 7000, and 6000 respectively. The value of each variable is written inside each cell; under the cells are their respective addresses in memory.



- c has type int*** and a value of 7000
- *c has type int ** and a value of 8000
- **c has type int * and a value of 9000
- ***c has type int and a value of 5

Pointer to constant and constant pointer

```
int x=5, z=7 ;
const int y=8;
const int *p; // here p is pointer to const int
int * const q=&x; // here q is constatnt pointer to int
int * const r; //error constant pointer must be initialized
p=&x; // valid since int * can be converted to const int *
*p=3; // error cannot modify const object
*q=3; //valid
q=&z; //error
int * const s=&y;// error  cannot convert const int * to int *
const int * const b= &x; //legal
const int * const c=&y; //legal
*b=2;//error cannot modify const object
 b=c; // error cannot modify const object
```

Static memory allocation and Dynamic memory allocation

If memory allocation is done at the time of compilation of the program then memory allocation is known as static memory allocation. For example memory allocation for variables and arrays are done at compilation of the program as size of variable and arrays are already known at the time of compilation of the program.

Dynamic memory allocation is the memory allocation required during execution of the program. For example if the amount of memory needed is determined by the value input by the user at run time.

Dynamic memory allocation using new operator

```
Int *a=new int(5);    /* new operator will create an int object somewhere in memory heap
                         and initialize the object with value 5 and returns address of the
                         object to pointer variable a  */
int n;
cin>>n;
Int *b=new int[n];    /*here new operator will create an array of 5 int and return the base
                         address of the allocated array to pointer variable b */
```

If the new operator fails to allocate memory then it returns NULL value which indicates that the required memory is not available for the requested instruction.

Note: We cannot create array of variable length for example

```
Int n;
Cin>>n;
Int a[n]; // error because we can use only constant integral value in array declaration
const int p=3;
int a[p]; //legal
```

Operators delete and delete[ ]

Since the necessity of dynamic memory is usually limited to specific moments within a program, once it is no longer needed it should be freed so that the memory becomes available again for other requests of dynamic memory. This is the purpose of the operator delete, whose format is:

delete pointer; //used to delete the memory allocated for single object
delete pointer[];//used to delete the memory allocated for array of objects

The value passed as argument to delete must be either a pointer to a memory block previously allocated with new, or a null pointer (in the case of a null pointer, delete produces no effect).

```
void main()
{int n;
 Int *q;
cout<<"enter no of integers required :";
cin>>n;
q=new int [n];
if(q==NULL)
        cout<<"memory could not be allocated";
else
        {for(int i=0;i<n;i++)
                q[i]=i+1;
        for(i=0;i<n;i++)
        cout<<q[i]<<" ";
        delete [ ]q;
        }
}
```

For n=3 and successful memory allocation the output of the program would be
1 2 3
Otherwise on failure of new operator the message "memory could not be allocated" would be the output.
It is always a good habit to check the pointer with NULL value to make sure that memory is allocated or not.

Memory leaks

If dynamically allocated memory has become unreachable (no pointer variable is pointing the allocated memory) then such situation is known as memory leak because neither it can be accessed nor it can be deleted from the memory heap by garbage collection. For example

```
void function1()
{
int x=5;
int *p;
p=new int [100]; // dynamic memory allocation for an array of 100 integers
p=&x;
}
```

In the above example the statement **p=new int [100]** assign the address of the dynamically allocated memory to p and the next statement **p=&x** assign the address of x to p therefore, after that the dynamically allocated memory become unreachable which cause memory leak.

Pointer to structure

```
struct emp
{
int empno;
char name[20];
float sal;
```

```
};
void main()
{
emp e1={5,"Ankit Singh", 30000.0};
emp *p=&e1;
cout<<p->empno<<"\t"<<p->name<<"\t"<<p->sal<<endl;
p->sal=p->sal + p->sal*3.0/100;
cout<<(*p).empno<<"\t"<<p->name<<"\t"<<(*p).sal;
}
```

Output is

```
5       Ankit Singh       30000.0
5       Ankit Singh       30900.0
```

Invoking Function by Passing the References :

When parameters are passed to the functions by reference, then the formal parameters become references (or aliases) to the actual parameters to the calling function.That means the called function does not create its own copy of original values, rather, it refers to theoriginal values by different names i.e. their references. For example the program of swapping two variables with reference method :

```
#include<iostream.h>
void main()
{
        void swap(int &, int &);
        int a = 5, b = 6;
        cout << "\n Value of a :" << a << " and b :" << b;
        swap(a, b);
        cout << "\n After swapping value of a :" << a << "and b :" << b;
}
void swap(int &m, int &n)
{
        int temp; temp = m;
        m = n;
        n = temp;
}
output :
Value of a : 5 and b : 6
After swapping value of a : 6 and b : 5
```

Invoking Function by Passing the Pointers:

When the pointers are passed to the function, the addresses of actual arguments in the calling function are copied into formal arguments of the called function. That means using the formal arguments (the addresses of original values) in the called function, we can make changing the actual arguments of the calling function. For example the program of swapping two variables with Pointers :

```
#include<iostream.h>
void main()
{
        void swap(int *m, int *n);
        int a = 5, b = 6;
        cout << "\n Value of a :" << a << " and b :" << b;
        swap(&a, &b);
```

```
        cout << "\n After swapping value of a :" << a << "and b :" << b;
}
void swap(int *m, int *n)
{
        int temp;
        temp = *m;
        *m = *n;
        *n = temp;
}
```

Input :
Value of a : 5 and b : 6

Function returning Pointers :

The way a function can returns an int, an float, it also returns a pointer. The general form of prototype
of a function returning a pointer would be
Type * function-name (argument list);

```
#include<iostream.h>
int *min(int &, int &);
void main()
{int a, b, *c;
cout << "\nEnter a :";
cin >> a;
cout << "\nEnter b :";
cin >> b;
c = min(a, b);
cout << "\n The minimum no is :" << *c;
}
int *min(int &x, int &y)
{
if (x < y )
    return (&x);
else
    return (&y);
}
```

Objects as Function arguments:

Objects are passed to functions in the same way as any other type of variable is passed.When it is said that objects are passed through the call-by-value, it means that the calledfunction creates a copy of the passed object.A called function receiving an object as a parameter creates the copy of the object without invoking the constructor. However, when the function terminates, it destroys this copy of the object by invoking its destructor function.

If you want the called function to work with the original object so that there is no need to create and destroy the copy of it, you may pass the reference of the object. Then the called function refers to the original object using its reference or alias.Also the object pointers are declared by placing in front of a object pointer's name. Class-name * object-pointer;

Eg. Student *stu;

The member of a class is accessed by the arrow operator (->) in object pointer

method.

Eg :

```
#include<iostream.h>
class Point
{
   int x, y;
        public :
            Point()
            {x = y = 0;}
            void getPoint(int x1, int y1)
            {x = x1; y = y1; }
            void putPoint()
            {
            cout << "\n Point : (" << x << ", " << y << ")";
            }};
        void main()
        {
        Point p1, *p2;
        cout << "\n Set point at 3, 5 with object";
        p1.getPoint(3,5);
        cout << "\n The point is :";
        p1.putPoint();
        p2 = &p1;
        cout << "\n Print point using object pointer :";
        p2->putPoint();
        cout << "\n Set point at 6,7 with object pointer";
        p2->getPoint(6,7);
        cout<< "\n The point is :";
        p2->putPoint();
        cout << "\n Print point using object :";
        p1.getPoint();
        }
```

If you make an object pointer point to the first object in an array of objects, incrementing the pointer would make it point to the next object in sequence.

```
student stud[5], *sp;
    sp = stud; // sp points to the first element (stud[0])of stud
    sp++;    // sp points to the second element (stud[1]) of stud
     sp + = 2;   // sp points to the fourth element (stud[3]) of stud
    sp--;        //sp points to the third element (stud[2]) of stud
```

this pointer

When an object of a class invokes a member function then a special pointer is implicitly passed to the member function which contains the address of the object by which the function is being called, this special pointer is known as ***this*** pointer. For example

```
#include<iostream.h>
class A
{int x;
```

```
 public:
void input()
{ cout<<"enter a number :";
  cin>>x;
}
void output()
{cout<<"address of the object is :"<<this<<endl;
 cout<<this->x<<"\t"<<x<<endl ; // both this->x and x will give the same value i.e. value of
 }
void main()
{
A a1, a2;
a1.input();
a2.input();
a1.output();
a2.output();
}
```

Output is

enter a number : 5

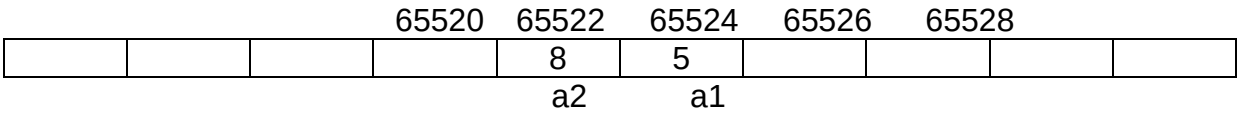
enter a number : 8

address of the object is : 65524 //assuming that a1 is allocated at memory location 1000

5 5

address of the object is : 65522 //assuming that a2 is allocated at memory location 998

8 8



Note: the ***this*** pointer does not exist outside the member function, i.e. we cannot define or access this pointer in main function because this pointer comes into existence only when a member function of a class is invoked by an object of the class.

2 Marks Questions

Exercise

1. Give the output of the following program:

```
void main()
{char *p = "School";
char c;
c = ++ *p ++;
cout<<c<<", "<<p<<endl;
cout<<p<<", "<<++*p- -<<", "<<++*p++;
}
```

2. Give the output of the following program:

```
void main()
{int x [ ] = {50, 40, 30, 20, 10}:
int *p, **q, *t;
p = x;
t = x + 1;
q = &t;
cout << *p <<", " << **q << "," << *t++;
```

}

3. Give the output of the following program (Assume all necessary header files are included):

```
void main( )
{char * x = “TajMahal”;
char c;
x=x+3 ;
c = ++ *x ++;
cout<<c<<”, “;
cout<< *x<<“, ”<<- -*x++<<- -x ;
}
```

4. Give the output of the following program (Assume all necessary header files are included):

```
void main( )
{char *x = “Rajasthan”;
char c;
c = (*(x+3))++ ;
cout<<c<<“, ”<<x<<“, ”<<sizeof(x)<< “, ”<<sizeof(x+3)<< “, ”<<strlen(x+3);
}
```

5. What will be the output of the program (Assume all necessary header files are included):

```
void print (char * p )
{int i=0;
while(*p)
        *p=*p++ + i++;
cout<<"value is "<<p-i+2<<endl;
}
void main( )
{char * x = "Mumbai";
print(x);
cout<<"new value is "<<x<<endl;
}
```

6. . Identify the errors if any. Also give the reason for errors.

```
#include<iostream.h>
void main()
{int b=4;
const int i =20;
const int * ptr=&i;
(*ptr)++;
ptr =&j;
cout<<*ptr;
}
```

7. Identify errors on the following code segment

```
void main()
{float c[ ] ={ 1.2,2.2,3.2,56.2};
float *k,*g;
k=c;
g=k+4;
k=k*2;
g=g/2;
k=k+g;
c=k;
cout<<”*k=”<<*k<<”*g=”<<*g;
```

```
}
```

8. What will be the output of the program (Assume all necessary header files are included):

```
void main( )
{int a[ ]={4,8,2,5,7,9,6}
int x=a+5,y=a+3;
cout<<++*x- -<<",”<<++*x- -<<",”<<++*y++<<",”<<++*y- -;
}
```

9. What will be the output of the program (Assume all necessary header files are included):

```
void main()
{int arr[ ] = {12, 23, 34, 45};
int *ptr = arr;
int val = *ptr ; cout << val << endl;
val = *ptr++; cout << val << endl;
val = *ptr; cout << val << endl;
val = *++ptr; cout << val << endl;
val = ++*ptr; cout << val << endl;
}
```

10. Differentiate between static and dynamic allocation of memory.

11. Identify and explain the error in the following program :

```
#include<iostream.h>
int main()
{int x[ ] = { 1, 2, 3, 4, 5 };
    for (int i = 0; i < 5; i++)
    {  cout << *x;
    x++;}
    return 0;
    }
```

12. Give the output of the following :

```
char *s = “computer”;
for (int x = strlen(s) – 1; x >= 0; x--)
{for(int y =0; y <= x; y++)       cout << s[y];
    cout << endl;
    }
```

3 Marks Questions

1.Give output of following code fragment assuming all necessary header files are included:

```
void main()
{char *msg = “Computer Science”;
for (int i = 0; i < strlen (msg); i++)
if (islower(msg[i]))
msg[i] = toupper (msg[i]);
else
if (isupper(msg[i]))
if( i % 2 != 0)
msg[i] = tolower (msg[i-1]);
else
msg[i--];
cout << msg << endl;
```

}

2. What will be the output of the program (Assume all necessary header files are included):

```
void main( )
{clrscr( );
int a =32;
int *ptr = &a;
char ch = 'A';
char *cho=&ch;
cho+=a; // it is simply adding the addresses.
*ptr + = ch;
cout<< a << "" <<ch<<endl;
}
```

3. What will be the output of the program (Assume all necessary header files are included):

```
void main( )
{char *a[ ]={"DELHI", "MUMBAI","VARANSAI"};
char **p;
p=a;
cout<<sizeof(a)<<", "<<sizeof(a[0])<<", "<<sizeof(p)<<endl;
cout<< p << ", "<<++*p<<", "<<*++p<<endl;
cout<< **a << ", " <<*(a+1)<<", " <<a[2]<<endl;
}
```

Data Structure

In Computer Science, a **data structure** is a particular way of storing and organizing data in a computer so that it can be used efficiently. Different kinds of data structures are suited to different kinds of applications, and some are highly specialized to specific tasks.

The data structure can be classified into following two types:
Simple Data Structure: These data structures are normally built from primitive data types like integers, floats, characters. For example arrays and structure.
Compound Data Structure: simple data structures can be combined in various ways to form more complex structure called compound structures. Linked Lists, Stack, Queues and Trees are examples of compound data structure.

Data Structure Arrays

Data structure array is defined as linear sequence of finite number of objects of same type with following set of operation:

- Creating : defining an array of required size
- Insertion: addition of a new data element in the in the array
- Deletion: removal of a data element from the array
- Searching: searching for the specified data from the array
- Traversing: processing all the data elements of the array
- Sorting : arranging data elements of the array in increasing or decreasing order
- Merging : combining elements of two similar types of arrays to form a new array of same type

In C++ an array can be defined as

```
Datatype arrayname[size];
```

Where size defines the maximum number of elements can be hold in the array. For example

```
float b[10];//b is an array which can store maximum 10 float values
int c[5];
```

Array initialization

```
Void main()
{
int b[10]={3,5,7,8,9};//
cout<<b[4]<<endl;
cout<<b[5]<<endl;
}
```

Output is
9
0

In the above example the statement int b[10]={3,5,7,8,9} assigns first 5 elements with the given values and the rest elements are initialized with 0. Since in C++ index of an array starts from 0 to size-1 so the expression b[4] denotes the 5th element of the array which is 9 and b[5] denotes 6th element which is initialized with 0.

3	5	7	8	9	0	0	0	0	0
b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]

Searching

We can use two different search algorithms for searching a specific data from an array

- Linear search algorithm
- Binary search algorithm

Linear search algorithm

In Linear search, each element of the array is compared with the given item to be searched for. This method continues until the searched item is found or the last item is compared.

```
#include<iostream.h>
int linear_search(int a[], int size, int item)
{
int i=0;
 While(i<size&& a[i]!=item)
    i++;
if(i<size)
   return i;//returns the index number of the item in the array
else
  return -1;//given item is not present in the array so it returns -1 since -1 is not a legal index
number
}
void main()
{
int b[8]={2,4,5,7,8,9,12,15},size=8;
int item;
cout<<"enter a number to be searched for";
cin>>item;
int p=linear_search(b, size, item); //search item in the array b
if(p==-1)
        cout<<item<<" is not present in the array"<<endl;
else
        cout<<item <<" is present in the array at index no "<<p;
}
```

In linear search algorithm, if the searched item is the first elements of the array then the loop terminates after the first comparison (best case), if the searched item is the last element of the array then the loop terminates after size time comparison (worst case) and if the searched item is middle element of the array then the loop terminates after size/2 time comparisons (average case). For large size array linear search not an efficient algorithm but it can be used for unsorted array also.

Binary search algorithm

Binary search algorithm is applicable for already sorted array only. In this algorithm, to search for the given item from the sorted array (in ascending order), the item is compared with the middle element of the array. If the middle element is equal to the item then index of the middle element is returned, otherwise, if item is less than the middle item then the item is present in first half segment of the array (i.e. between 0 to middle-1), so the next iteration will continue for first half only, if the item is larger than the middle element then the item is present in second half of the array (i.e. between middle+1 to size-1), so the next iteration will continue for second half segment of the array only. The same process continues until either the item is found (search successful) or the segment is reduced to the single element and still the item is not found (search unsuccessful).

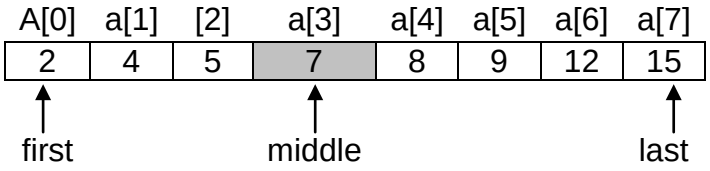
```
#include<iostream.h>
int binary_search(int a[ ], int size, int item)
{
int first=0,last=size-1,middle;
while(first<=last)
        {
        middle=(first+last)/2;
        if(item==a[middle])
                return middle; // item is found
        else if(item< a[middle])
```

```
            last=middle-1; //item is present in left side of the middle element
        else
            first=middle+1; // item is present in right side of the middle element
        }
return -1; //given item is not present in the array, here, -1 indicates unsuccessful search
}
void main()
{
int b[8]={2,4,5,7,8,9,12,15},size=8;
int item;
cout<<"enter a number to be searched for";
cin>>item;
int p=binary_search(b, size, item); //search item in the array b
if(p==-1)
        cout<<item<<" is not present in the array"<<endl;
else
        cout<<item <<" is present in the array at index no "<<p;
}
```

Let us see how this algorithm work for item=12

Initializing first =0 ; last=size-1; where size=8

Iteration 1



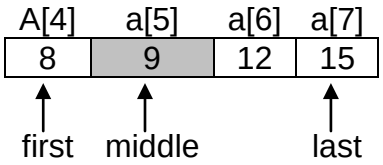
First=0, last=7

middle=(first+last)/2=(0+7)/2=3 // note integer division 3.5 becomes 3

value of a[middle] i.e. a[3] is 7

7<12 then first= middle+1 i.e. 3 + 1 =4

iteration 2



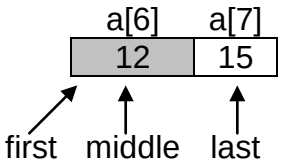
first=4, last=7

middle=(first+last)/2=(4+7)/2=5

value of a[middle] i.e. a[5] is 9

9<12 then first=middle+1;5+1=6

iteration 3



first=6,last=7

middle=(first+last)/2 = (6+7)/2=6

value of a[middle] i.e. a[6] is 12 which is equal to the value of item being search i.e.12

As a successful search the function binary_search() will return to the main function with value 6 as index of 12 in the given array. In main function p hold the return index number.

Note that each iteration of the algorithm divides the given array in to two equal segments and the only one segment is compared for the search of the given item in the next iteration. For a given array of size N= 2^n elements, maximum n number of iterations are required to make sure whether the given item is present in the given array or not, where as the linear requires maximum 2^n number of iteration. For example, the number of iteration required to search an item in the given array of 1000 elements, binary search requires maximum 10 (as $1000 \approx 2^{10}$) iterations where as linear search requires maximum 1000 iterations.

Inserting a new element in an array

We can insert a new element in an array in two ways

- If the array is unordered, the new element is inserted at the end of the array
- If the array is sorted then the new element is added at appropriate position without altering the order. To achieve this, all elements greater than the new element are shifted. For example, to add 10 in the given array below:

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
2	4	5	7	8	11	12	15	

Original array

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
2	4	5	7	8		11	12	15

Elements greater than 10 shifted to create free place to insert 10

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]
2	4	5	7	8	10	11	12	15

Array after insertion

Following program implement insertion operation for sorted array

```
#include<iostream.h>
void insert(int a[ ], int &n,  int item) //n is the number of elements already present in the array
{
int i=n-1;
while (i>=0 && a[i]>item)
        {
        a[i+1]=a[i];  // shift the i^th element one position towards right
         i--;
        }
a[i+1]=item;    //insertion of item at appropriate place
n++; //after insertion, number of elements present in the array is increased by 1
}
void main()
{int a[10]={2,4,5,7,8,11,12,15},n=8;
 int i=0;
 cout<<"Original array is:\n";
 for(i=0;i<n;i++)
        cout<<a[i]<<", ";
```

```
insert(a,n,10);
cout<<"\nArray after inserting 10 is:\n";
for(i=0; i<n; i++)
        cout<<a[i]<<", ";
}
```

Output is
Original array is:
2, 4, 5, 7, 8, 11, 12, 15
Array after inserting 10 is:
2, 4, 5, 7, 8, 10, 11, 12, 15

Deletion of an item from a sorted array

In this algorithm the item to be deleted from the sorted array is searched and if the item is found in the array then the element is removed and the rest of the elements are shifted one position toward left in the array to keep the ordered array undisturbed. Deletion operation reduces the number of elements present in the array by1. For example, to remove 11 from the given array below:

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]
2	4	5	7	8	11	12	15

Original array

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]
2	4	5	7	8		12	15

Element removed

a[0]	a[1]	[2]	a[3]	a[4]	a[5]	a[6]	a[7]
2	4	5	7	8	12	15	

Array after shifting the rest element

Following program implement deletion operation for sorted array

```
#include<iostream.h>
void delete_item(int a[ ], int &n,  int item) //n is the number of elements already present in the array
{int i=0;
while(i<n && a[i]<item)
        i++;
if (a[i]==item)  // given item is found
        {while (i<n)
                {a[i]=a[i+1];  // shift the (i+1)th element one position towards left
                 i++;
                }
        cout<<"\n Given item is successfully deleted";
        n--;
        }
 else
        cout<<"\n Given item is not found in the array";
}
void main()
{int a[10]={2,4,5,7,8,11,12,15},n=8;
 int i=0;
```

```
 cout<<"Original array is :\n";
 for(i=0;i<n;i++)
        cout<<a[i]<<", ";
delete_item(a,n,11);
cout<<"\nArray after deleting 11 is:\n";
for(i=0; i<n; i++)
        cout<<a[i]<<", ";
}
```

Output is
Original array is:
2, 4, 5, 7, 8, 11, 12, 15
Given item is successfully deleted
Array after deleting 11 is:
2, 4, 5, 7, 8, 12, 15

Traversal

Processing of all elements (i.e. from first element to the last element) present in one-dimensional array is called traversal. For example, printing all elements of an array, finding sum of all elements present in an array.

```
#include<iostream.h>
void print_array(int a[ ], int n) //n is the number of elements present in the array
{int i;
cout<<"\n Given array is :\n";
for(i=0; i<n; i++)
        cout<<a[i]<<", ";
}
int sum(int a[ ], int n)
{int i,s=0;
for(i=0; i<n; i++)
        s=s+a[i];
return s;
}
void main()
{int b[10]={3,5,6,2,8,4,1,12,25,13},n=10;
int i, s;
print_array(b,n);
s=sum(b,n);
cout<<"\n Sum of all elements of the given array is : "<<s;
}
```

Output is
Given array is
3, 5, 6, 2, 8, 4, 1, 12, 25, 13
Sum of all elements of the given array is : 79

Sorting

The process of arranging the array elements in increasing (ascending) or decreasing (descending) order is known as sorting. There are several sorting techniques are available e.g. selection sort, insertion sort, bubble sort, quick sort, heap short etc. But in CBSE syllabus only selection sort, insertion sort, bubble sort are specified.

Selection Sort

The basic idea of a selection sort is to repeatedly select the smallest element in the remaining unsorted array and exchange the selected smallest element with the first element

of the unsorted array. For example, consider the following unsorted array to be sorted using selection sort

Original array

0	1	2	3	4	5	6
8	5	9	3	16	4	7

iteration 1 : Select the smallest element from unsorted array which is 3 and exchange 3 with the first element of the unsorted array i.e. exchange 3 with 8. After iteration 1 the element 3 is at its final position in the array.

0	1	2	3	4	5	6
3	5	9	8	16	4	7

Iteration 2: The second pass identify 4 as the smallest element and then exchange 4 with 5

0	1	2	3	4	5	6
3	4	9	8	16	5	7

Iteration 3: The third pass identify 5 as the smallest element and then exchange 5 with 9

0	1	2	3	4	5	6
3	4	5	8	16	9	7

Iteration 4: The third pass identify 7 as the smallest element and then exchange 7 with 8

0	1	2	3	4	5	6
3	4	5	7	16	9	8

Iteration 5: The third pass identify 8 as the smallest element and then exchange 8 with 16

0	1	2	3	4	5	6
3	4	5	7	8	9	16

Iteration 6: The third pass identify 9 as the smallest element and then exchange 9 with 9 which makes no effect.

0	1	2	3	4	5	6
3	4	5	7	8	9	16

The unsorted array with only one element i.e. 16 is already at its appropriate position so no more iteration is required. Hence to sort n numbers, the number of iterations required is n-1, where in each next iteration, the number of comparison required to find the smallest element is decreases by 1 as in each pass one element is selected from the unsorted part of the array and moved at the end of sorted part of the array . For n=7 the total number of comparison required is calculated as

Pass1: 6 comparisons i.e. (n-1)
Pass2: 5 comparisons i.e. (n-2)
Pass3: 4 comparisons i.e. (n-3)
Pass4: 3 comparisons i.e. (n-4)
Pass5: 2 comparisons i.e. (n-5)
Pass6: 1 comparison i.e. (n-6)=(n-(n-1))
Total comparison for n=(n-1)+(n-2)+(n-3)+ …….+(n-(n-1))= n(n-1)/2
7=6+5+4+3+2+1=7*6/2=21;

Note: For given array of n elements, selection sort always executes n(n-1)/2 comparison statements irrespective of whether the input array is already sorted(best case), partially sorted(average case) or totally unsorted(i.e. in reverse order)(worst case).

Program:

```
#include<iostream.h>
void select_sort(int a[ ], int n) //n is the number of elements present in the array
{int i, j, p, small;
```

```
for(i=0;i<n-1;i++)
        {small=a[i]; // initialize small with the first element of unsorted part of the array
        p=i;            // keep index of the smallest number of unsorted part of the array in p
        for(j=i+1; j<n; j++) //loop for selecting the smallest element form unsorted array
                {if(a[j]<small)
                        {small=a[j];
                        p=j;
                        }
                }// end of inner loop----------
         //----------exchange the smallest element with ith element----------
        a[p]=a[i];
        a[i]=small;
        //-----------end of exchange----------
        }
}//end of function
void main( )
{int a[7]={8,5,9,3,16,4,7},n=7,i;
cout<<"\n Original array is :\n";
for(i=0;i<n;i++)
        cout<<a[i]<<", ";
select_sort(a,n);
cout<<"\nThe sorted array is:\n";
for(i=0; i<n; i++)
        cout<<a[i]<<", ";
}
```

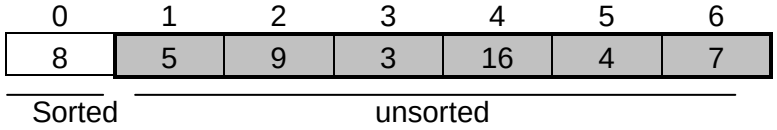
Output is
Original array is
8, 5, 9, 3, 16, 4, 7
The sorted array is
3, 4, 5, 7, 8, 9, 16

Insertion Sort

Insertion sort algorithm divides the array of n elements in to two subparts, the first subpart contain a[0] to a[k] elements in sorted order and the second subpart contain a[k+1] to a[n] which are to be sorted. The algorithm starts with only first element in the sorted subpart because array of one element is itself in sorted order. In each pass, the first element of the unsorted subpart is removed and is inserted at the appropriate position in the sorted array so that the sorted array remain in sorted order and hence in each pass the size of the sorted subpart is increased by 1 and size of unsorted subpart is decreased by 1. This process continues until all n-1 elements of the unsorted arrays are inserted at their appropriate position in the sorted array.

For example, consider the following unsorted array to be sorted using selection sort

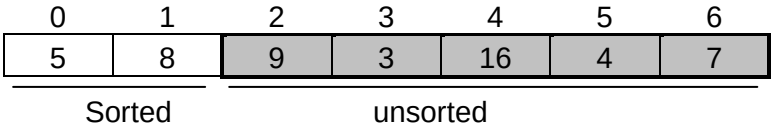
Original array



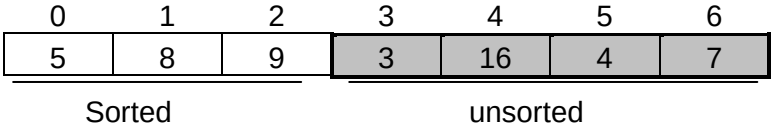
Initially the sorted subpart contains only one element i.e. 8 and the unsorted subpart contains n-1 elements where n is the number of elements in the given array.

Iteration1: To insert first element of the unsorted subpart i.e. 5 into the sorted subpart, 5 is compared with all elements of the sorted subpart starting from rightmost element to the leftmost element whose value is greater than 5, shift all elements of the sorted subpart whose value is greater than 5 one position

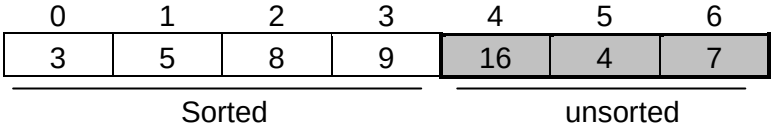
towards right to create an empty place at the appropriate position in the sorted array, store 5 at the created empty place, here 8 will move from position a[0] to a[1] and a[0] is filled by 5. After first pass the status of the array is:



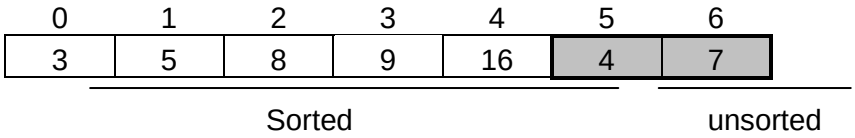
Iteration2: In second pass 9 is the first element of the unsorted subpart, 9 is compared with 8, since 8 is less than 9 so no shifting takes place and the comparing loop terminates. So the element 9 is added at the rightmost end of the sorted subpart. After second pass the status of the array is:



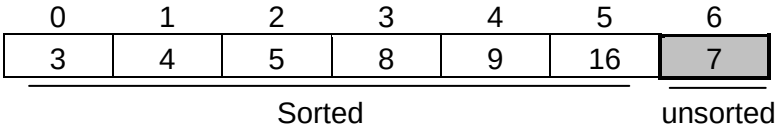
Iteration3: in third pass 3 is compared with 9, 8 and 5 and shift them one position towards right and insert 3 at position a[0]. After third pass the status of the array is:



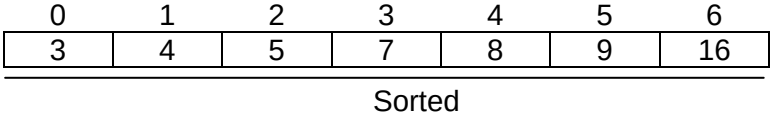
Iteration4: in forth pass 16 is greater than the largest number of the sorted subpart so it remains at the same position in the array. After fourth pass the status of the array is:



Iteration5: in fifth pass 4 is inserted after 3. After third pass the status of the array is:



Iteration6: in sixth pass 7 is inserted after 5. After fifth pass the status of the array is:



Insertion sort take advantage of sorted(best case) or partially sorted(average case) array because if all elements are at their right place then in each pass only one comparison is required to make sure that the element is at its right position. So for n=7 only 6 (i.e. n-1) iterations are required and in each iteration only one comparison is required i.e. total number of comparisons required= (n-1)=6 which is better than the selection sort (for sorted array selection sort required n(n-1)/2 comparisons). Therefore insertion sort is best suited for sorted or partially sorted arrays.

Program to implement insert sort:

```
#include<iostream.h>
void insert_sort(int a[ ],int n) //n is the no of elements present in the array
```

```
{int i, j,p;
for (i=1; i<n; i++)
        {p=a[i];
        j=i-1;
        //inner loop to shift all elements of sorted subpart one position towards right
        while(j>=0&&a[j]>p)
                {
                a[j+1]=a[j];
                j--;
                }
        //---------end of inner loop
        a[j+1]=p;       //insert p in the sorted subpart
        }
}
void main( )
{
int a[7]={8,5,9,3,16,4,7},n=7,i;
cout<<"\n Original array is :\n";
for(i=0;i<n;i++)
        cout<<a[i]<<", ";
insert_sort(a,n);
cout<<"\nThe sorted array is:\n";
for(i=0; i<n; i++)
        cout<<a[i]<<", ";
}
```

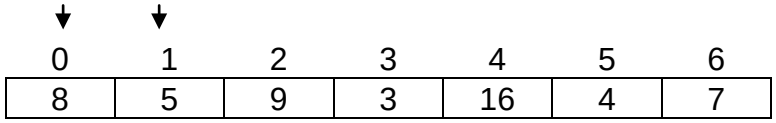
Output is
Original array is
8, 5, 9, 3, 16, 4, 7
The sorted array is
3, 4, 5, 7, 8, 9, 16

Bubble Sort

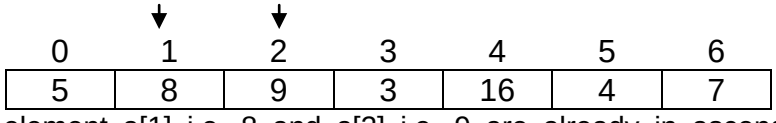
Bubble sort compares a[i] with a[i+1] for all i=0..n-2, if a[i] and a[i+1] are not in ascending order then exchange a[i] with a[i+1] immediately. After each iteration all elements which are not at their proper position move at least one position towards their right place in the array. The process continues until all elements get their proper place in the array (i.e. algorithm terminates if no exchange occurs in the last iteration)

For example, consider the following unsorted array to be sorted using selection sort

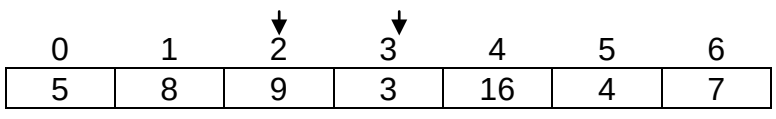
Original array



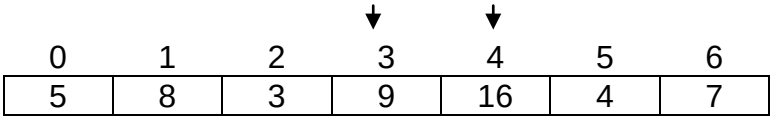
Iteration1: The element a[0] i.e. 8 is compared with a[1] i.e. 5, since 8>5 therefore exchange 8 with 5.



The element a[1] i.e. 8 and a[2] i.e. 9 are already in ascending order so no exchange required

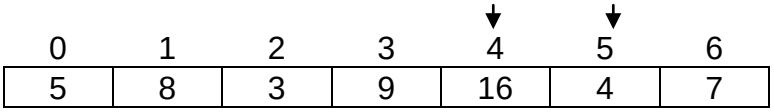


The element a[2] i.e. 9 and a[3] i.e. 3 are not in ascending order so exchange a[2] with a[3]



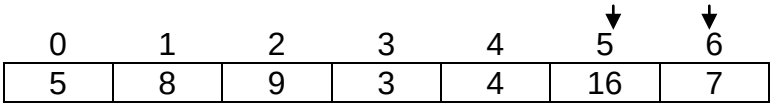
0	1	2	3	4	5	6
5	8	3	9	16	4	7

The element a[3] i.e. 9 and a[4] i.e. 16 are in ascending order so no exchange required



0	1	2	3	4	5	6
5	8	3	9	16	4	7

The element a[4] i.e. 16 and a[5] i.e. 4 are not in ascending order so exchange a[4] with a[5]



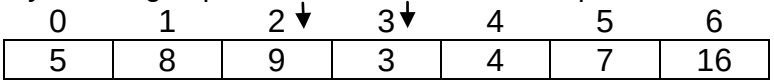
0	1	2	3	4	5	6
5	8	9	3	4	16	7

The element a[5] i.e. 16 and a[6] i.e. 7 are not in ascending order so exchange a[5] with a[6]

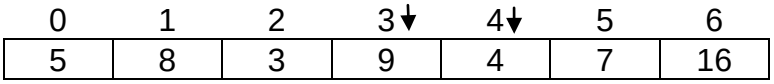
0	1	2	3	4	5	6
5	8	9	3	4	7	16

Since in iteration1 some elements were exchanged with each other which shows that array was not sorted yet, next iteration continues. The algorithm will terminate only if the last iteration do not process any exchange operation which assure that all elements of the array are in proper order.

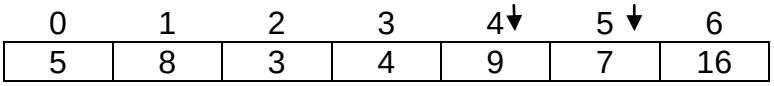
Iteration2: only exchange operations are shown in each pass



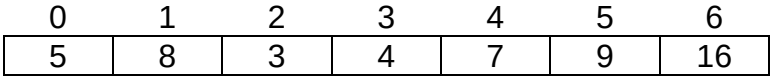
0	1	2	3	4	5	6
5	8	9	3	4	7	16



0	1	2	3	4	5	6
5	8	3	9	4	7	16



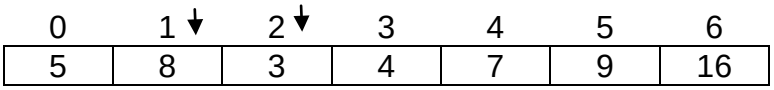
0	1	2	3	4	5	6
5	8	3	4	9	7	16



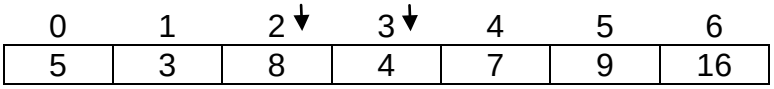
0	1	2	3	4	5	6
5	8	3	4	7	9	16

In iteration 2 some exchange operations were processed, so, at least one more iteration is required to assure that array is in sorted order.

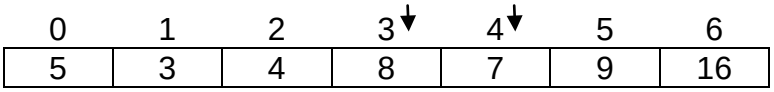
Iteration3:



0	1	2	3	4	5	6
5	8	3	4	7	9	16



0	1	2	3	4	5	6
5	3	8	4	7	9	16



0	1	2	3	4	5	6
5	3	4	8	7	9	16

0	1	2	3	4	5	6
5	3	4	7	8	9	16

Iteration4:

0	1	2	3	4	5	6
5	3	4	7	8	9	16

0	1	2	3	4	5	6
3	5	4	7	8	9	16

0	1	2	3	4	5	6
3	4	5	7	8	9	16

Iteration5:

0	1	2	3	4	5	6
3	4	5	7	8	9	16

In iteration 5 no exchange operation executed because all elements are already in proper order therefore the algorithm will terminate after 5th iteration.

Program to implement insert sort:

```
#include<iostream.h>
void bubble_sort(int a[ ],int n) //n is the no of elements present in the array
{int i, t,flag;
      do{flag=0;
         for(i=0;i<n-1;i++)
                  {if(a[i]>a[i+1])            //interchange a[i] with a[i+1] if a[i]>a[i+1]
                           {t=a[i];
                            a[i]=a[i+1];
                            a[i+1]=t;
                           flag=1;
                           }
         }while(flag==1);  //loop will continue if any interchange is performed
}
void main( )
{
int a[7]={8,5,9,3,16,4,7},n=7,i;
cout<<"\n Original array is :\n";
for(i=0;i<n;i++)
         cout<<a[i]<<", ";
bubble_sort(a,n);
cout<<"\nThe sorted array is:\n";
for(i=0; i<n; i++)
         cout<<a[i]<<", ";
}
```

Output is
Original array is
8, 5, 9, 3, 16, 4, 7
The sorted array is
3, 4, 5, 7, 8, 9, 16

Merging of two sorted arrays into third array in sorted order

Algorithm to merge arrays a[m](sorted in ascending order) and b[n](sorted in descending order) into third array C[n+m] in ascending order.

```
#include<iostream.h>
```

```
Merge(int a[ ], int m, int b[n], int c[ ])// m is size of array a and n is the size of array b
{int i=0; //    i points to the smallest element of the array a which is at index 0
 int j=n-1;// j points to the smallest element of the array b which is at the index m-1 since b is
              // sortet in descending order
int k=0;  //k points to the first element of the array c
while(i<m&&j>=0)
     {if(a[i]<b[j])
              c[k++]=a[i++]; // copy from array a into array c and then increment i and k
      else
            c[k++]=b[j--]; // copy from array b into array c and then decrement j and increment k

    }
while(i<m)          //copy all remaining elements of array a
      c[k++]=a[i++];
while(j>=0)        //copy all remaining elements of array b
      c[k++]=b[j--];
}
void main()
{int a[5]={2,4,5,6,7},m=5; //a is in ascending order
int b[6]={15,12,4,3,2,1},n=6; //b is in descending order
int c[11];
merge(a, m, b, n, c);
cout<<"The merged array is :\n";
for(int i=0; i<m+n; i++)
          cout<<c[i]<", ";
}
```

Output is
The merged array is:
1, 2, 2, 3, 4, 4, 5, 6, 7, 12, 15

Two dimensional arrays

In computing, **row-major order** and **column-major order** describe methods for storing multidimensional arrays in linear memory. Following standard matrix notation, rows are identified by the first index of a two-dimensional array and columns by the second index. Array layout is critical for correctly passing arrays between programs written in different languages. Row-major order is used in C, C++; column-major order is used in Fortran and MATLAB.

Row-major order

In row-major storage, a multidimensional array in linear memory is accessed such that rows are stored one after the other. When using row-major order, the difference between addresses of array cells in increasing rows is larger than addresses of cells in increasing columns. For example, consider this 2×3 array:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

An array declared in C as
`int A[2][3] = { {1, 2, 3}, {4, 5, 6} };`
would be laid out contiguously in linear memory as:

```
1 2 3 4 5 6
```

To traverse this array in the order in which it is laid out in memory, one would use the following nested loop:

```
for (i = 0; i < 2; i++)
   for (j = 0; j < 3; j++)
      cout<<A[i][j];
```

The difference in offset from one column to the next is 1*sizeof(type) and from one row to the next is 3*sizeof(type). The linear offset from the beginning of the array to any given element A[row][column] can then be computed as:

offset = row*NUMCOLS + column

Address of element A[row][column] can be computed as:

Address of A[row][column]=**base address of A + (row*NUMCOLS + column)* sizeof (type)**

Where **NUMCOLS** is the number of columns in the array.

The above formula only works when using the C, C++ convention of labeling the first element 0. In other words, row 1, column 2 in matrix A, would be represented as A[0][1]
Note that this technique generalizes, so a 2×2×2 array looks like:

```
int A[2][2][2] = {{{1,2}, {3,4}}, {{5,6}, {7,8}}};
```

and the array would be laid out in linear memory as:

```
1 2 3 4 5 6 7 8
```

Example 1.
For a given array A[10][20] is stored in the memory along the row with each of its elements occupying 4 bytes. Calculate address of A[3][5] if the base address of array A is 5000.

Solution:

For given array A[M][N] where M=Number of rows, N =Number of Columns present in the array
address of A[I][J]= base address+(I * N + J)*sizeof(type)
here M=10, N=20, I=3, J=5, sizeof(type)=4 bytes

address of A[3][5] = 5000 + (3 * 20 + 5) * 4
= 5000 + 65*4=5000+260=5260

Example 2.
An array A[50][20] is stored in the memory along the row with each of its elements occupying 8 bytes. Find out the location of A[5][10], if A[4][5] is stored at 4000.

Solution:

Calculate base address of A i.e. address of A[0][0]
For given array A[M][N] where M=Number of rows, N =Number of Columns present in the array
address of A[I][J]= base address+(I * N + J)*sizeof(type)
here M=50, N=20, sizeof(type)=8, I=4, J=5
address of A[4][5] = base address + (4*20 +5)*8
4000 = base address + 85*8
Base address= 4000-85*8= 4000-680=3320
Now to find address of A[5][10]
here M=50, N=20, sizeof(type)=8, I=5, J=10
Address of A[5][10] = base address +(5*20 + 10)*8
=3320 + 110*8 = 3320+880 = 4200

Column-major order is a similar method of flattening arrays onto linear memory, but the columns are listed in sequence. The programming languages Fortran, MATLAB, use column-major ordering. The array

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

if stored contiguously in linear memory with column-major order would look like the following:

```
1 4 2 5 3 6
```

The memory offset could then be computed as:
offset = row + column*NUMROWS

Address of element A[row][column] can be computed as:

Address of A[row][column]=**base address of A + (column*NUMROWS +rows)* sizeof (type)**

Where **NUMROWS** represents the number of rows in the array in this case, 2.

Treating a row-major array as a column-major array is the same as transposing it. Because performing a transpose requires data movement, and is quite difficult to do in-place for non-square matrices, such transpositions are rarely performed explicitly. For example, software libraries for linear algebra, such as the BLAS, typically provide options to specify that certain matrices are to be interpreted in transposed order to avoid the necessity of data movement

Example1.
For a given array A[10][20] is stored in the memory along the column with each of its elements occupying 4 bytes. Calculate address of A[3][5] if the base address of array A is 5000.
Solution:
For given array A[M][N] where M=Number of rows, N =Number of Columns present in the array
Address of A[I][J]= base address + (J * M + I)*sizeof(type)
here M=10, N=20, I=3, J=5, sizeof(type)=4 bytes

Address of A[3][5] = 5000 + (5 * 10 + 3) * 4
= 5000 + 53*4 = 5000+212 =5212

Example2.
An array A[50][20] is stored in the memory along the column with each of its elements occupying 8 bytes. Find out the location of A[5][10], if A[4][5] is stored at 4000.
Solution:
Calculate base address of A i.e. address of A[0][0]
For given array A[M][N] where M=Number of rows, N =Number of Columns present in the array
address of A[I][J]= base address+(J * M + I)*sizeof(type)
here M=50, N=20, sizeof(type)=8, I=4, J=5
address of A[4][5] = base address + (5 * 50 +4)*8
4000 = base address + 254*8
Base address= 4000-254*8= 4000-2032=1968
Now to find address of A[5][10]
here M=50, N=20, sizeof(type)=8, I =5, J=10
Address of A[5][10] = base address +(10*50 + 10)*8
=1968 + 510*8 = 1968+4080 = 6048

4 Marks Questions

1. Write a function in C++ which accepts an integer array and its size as arguments and replaces elements having even values with its half and elements having odd values with twice its value
2. Write a function in C++ which accepts an integer array and its size as argument and exchanges the value of first half side elements with the second half side elements of the array.
Example: If an array of eight elements has initial content as 2,4,1,6,7,9,23,10. The function should rearrange the array as 7,9,23,10,2,4,1,6.
3. Write a function in c++ to find and display the sum of each row and each column of 2 dimensional array. Use the array and its size as parameters with int as the data type of the array.
4. Write a function in C++, which accepts an integer array and its size as parameters and rearrange the array in reverse. Example if an array of five members initially contains the elements as 6,7,8,13,9,19
Then the function should rearrange the array as 19,9,13,8,7,6
5. Write a function in C++, which accept an integer array and its size as arguments and swap the elements of every even location with its following odd location. Example : if an array of nine elements initially contains the elements as 2,4,1,6,5,7,9,23,10 Then the function should rearrange the array as 4,2,6,1,7,5,23,9,10
6. Write a function in C++ which accepts an integer array and its size as arguments and replaces elements having odd values with thrice and elements having even values with twice its value. Example: If an array of five elements initially contains the elements 3,4,5,16,9
Then the function should rearrange the content of the array as 9,8,15,32,27
7. Write function SORTPOINTS() in c++ to sort an array of structure Game in descending order of points using Bubble Sort Note: Assume the following definition of structure Game

```
struct Game
{long PNo; // Player Number
char PName[20];
long points;
};
```

8. Write a c++ function to shift all the negative numbers to left and positive number in the right side.
9. Define a function SWPCOL() in C++ to swap (interchange) the first column elements with the last column elements, for a two dimensional array passed as the argument of the function.
Example : if the two dimensional array contains

2 1 4 9
1 3 7 7
5 8 6 3
7 2 1 2
After swapping of the content of 1st and last column, it should be
9 1 4 2
7 3 7 1
3 8 6 5
2 2 1 7

13. Write a function which accept 2D array of integers and its size as arguments and displays the sum of elements which lie on diagonals.

[Assuming the 2D array to be a square matrix with odd dimension ie 3 x 3 , 4 x 4 etc]
Example of the array content is
5 4 3
6 7 8
1 2 9
Output through the function should be
Diagonal One Sum : 21
Diagonal Two: 11

15. Write a user defined function named upperhalf() which takes a 2D array A, with size n rows and n cols as arguments and print the upper half of the matrix

16. Write a user defined function lowerhalf() which takes a 2D array, with size n rows and n cols as argument and prints the lower half of the matrix

17. Write the function to find the largest and second largest number from a two dimensional array. The function should accept the array and its size as argument.

18. Write a function in C++ to merge the contents of two sorted arrays A & B into third array C. Assuming array A is sorted in ascending order, B is sorted in descending order, the resultant array is required to be in ascending order.

19. An array arr[40][30] stored in the memory along the column with each of the element occupying 4 bytes. Find out the base address and address of the element arr[20][15], if an element arr[15][10] is stored at the memory location 7200.

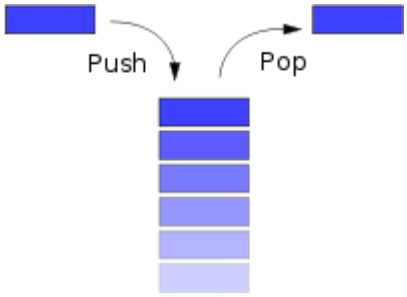
20. An array s[50][10] stored in the memory along the row with each element occupying 2 bytes. Find out the address of the location s[20][50] if the location s[10][25] stored at the address 10000.

Stack, Queues using Arrays and Linked List

Stack

In computer science, a stack is a last in, first out (LIFO) data structure. A stack can is characterized by only two fundamental operations: *push* and *pop*. The push operation adds an item to the top of the stack. The pop operation removes an item from the top of the stack, and returns this value to the caller.

A stack is a *restricted data structure*, because only a small number of operations are performed on it. The nature of the pop and push operations also mean that stack elements have a natural order. Elements are removed from the stack in the reverse order to the order of their addition: therefore, the lower elements are those that have been on the stack the longest. One of the common uses of stack is in function call.



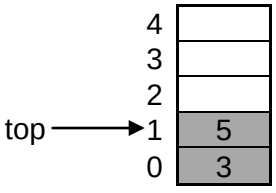
Stack using array

```
#include<iostream.h>
const int size=5
class stack
{int a[size];      //array a can store maximum 5 item of type int of the stack
int top;           //top will point to the last item pushed onto the stack
public:
stack(){top=-1;}         //constructor to create an empty stack, top=-1 indicate that no item is
                         //present in the array
void push(int item)
{If(top==size-1)
        cout<<"stack is full, given item cannot be added";
else
       a[++top]=item; //increment top by 1 then item at new position of the top in the array a
}
int pop()
{If (top==-1)
     {out<<"Stack is empty ";
      return -1; //-1 indicates empty stack
     }
else
     return a[top--];//return the item present at the top of the stack then decrement top by 1
}
void main()
{ stack s1;
  s1.push(3);
  s1.push(5);
cout<<s1.pop()<<endl;
cout<<s1.pop()<<endl;
cout<<s1.pop();
}
```

Output is

5

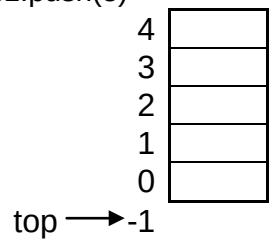
3



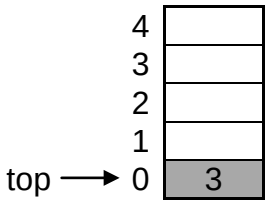
Stack is empty -1

In the above program the statement **stack s1** creates s1 as an empty stack and the constructor initialize top by -1.

Initially stack is empty

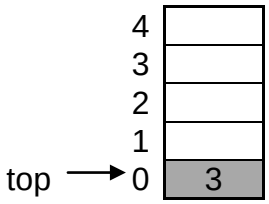


stack after s1.push(3)

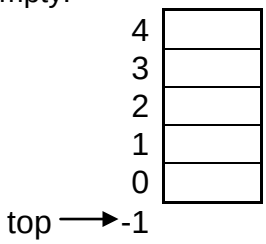


stack after s1.push(5)

After first s1.pop() statement, the item 5 is removed from the stack and top moves from 1 to 0



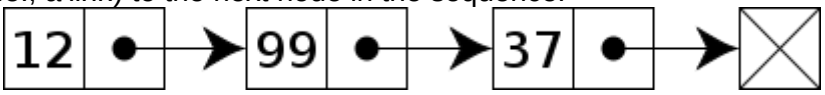
After second s1.pop() statement, the item 3 is removed from stack and top moves from 0 to -1 which indicates that now stack is empty.



After third s1.pop() statement the pop function display error message “stack is empty” and returns -1 to indicating that stack is empty and do not change the position of top of the stack.

Stack using Linked list

In Computer Science, a **linked list** (or more clearly, "singly-linked list") is a data structure that consists of a sequence of nodes each of which contains data and a pointer which points (i.e., a *link*) to the next node in the sequence.



A linked list whose nodes contain two fields: an integer value and a link to the next node

The main benefit of a linked list over a conventional array is that the list elements can easily be added or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory or on disk .Stack using linked lists allow insertion and removal of nodes only at the position where the pointer top is pointing to.

Stack implementation using linked list

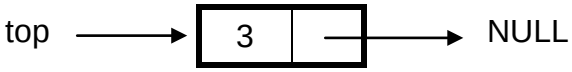
```
#include<iostream.h>
struct node
```

```
{
Int item; //data that will be stored in each node
node * next; //pointer which contains address of another node
}; //node is a self referential structure which contains reference of another object type node

class stack
{node *top;
public:
stack()  //constructor to create an empty stack by initializing top with NULL
{ top=NULL; }
void push(int item);
int pop();
~stack();
};
void stack::push(int item)    //to insert a new node at the top of the stack
{node *t=new node; //dynamic memory allocation for a new object of node type
if(t==NULL)
        cout<<"Memory not available, stack is full";
else
        {t->item=item;
        t->next=top; //newly created node will point to the last inserted node or NULL if
                        //stack is empty
        top=t;          //top will point to the newly created node
        }
}

int stack::pop()//to delete the last inserted node(which is currently pointed by the top)
{if(top==NULL)
        {cout<<"Stack is empty \n";
        return 0; // 0 indicating that stack is empty
        }
else
        {node *t=top; //save the address of top in t
        int r=top->item;  //store item of the node currently pointed by top
        top=top->next;  // move top from last node to the second last node
        delete t; //remove last node of the stack from memory
        return r;
        }
}
stack::~stack()//de-allocated all undeleted nodes of the stack when stack goes out of scope
{node *t;
while(top!=NULL)
        {t=top;
        top=top->next;
        delete t;
        }
};

void main()
{ stack s1;
  s1.push(3);
  s1.push(5);
  s1.push(7);
  cout<<s1.pop()<<endl;
cout<<s1.pop()<<endl;
```

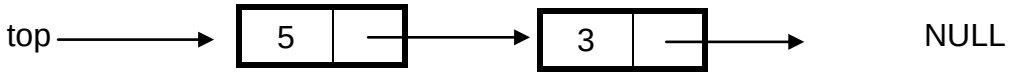
```
cout<<s1.pop()<<endl;
cout<<s1.pop()<<endl;
}
```
Output is
7
5
3
Stack is empty 0

In the above program the statement *stack s1;* invokes the constructor stack() which create an empty stack object s1 and initialize top with NULL.

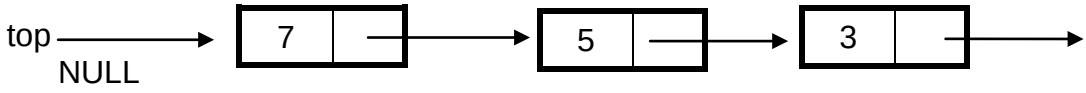
After statement s1.push(3) the stack become



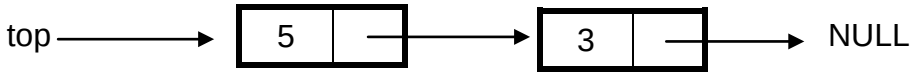
After statement s1.push(5) the stack become



After statement s1.push(7) the stack become

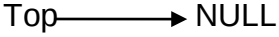


After the first s1.pop() statement the node currently pointed by top (i.e. node containing 7) is deleted from the stack, after deletion the status of stack is



After the second s1.pop() statement the node currently pointed by top (i.e. node containing 5) is deleted from the stack, after deletion the status of stack is

After the third s1.pop() statement the node currently pointed by top (i.e. node containing 3) is deleted from the stack, after deletion the stack become empty i.e.



After the fourth s1.pop() statement, the error message “stack is empty“ displayed and the pop() function return 0 to indicate that stack is empty.

Application of stacks in infix expression to postfix expression conversion

Infix expression	**operand1 operator operand2** for example **a+b**
Postfix expression	**operand1 operand2 operator** for example **ab+**
Prefix expression	**operator operand1 operand2** for example **+ab**

Some example of infix expression and their corresponding postfix expression

Infix expression	postfix expression
a*(b-c)/e	abc-*e/

(a+b)*(c-d)/e	ab+cd-*e/
(a+b*c)/(d-e)+f	abc*+de-/f+

Algorithm to convert infix expression to postfix expression using stack

Let the infix expression INEXP is to be converted in to equivalent postfix expression POSTEXP. The postfix expression POSTEXP will be constructed from left to right using the operands and operators (except "(", and ")") from INEXP. The algorithm begins by pushing a left parenthesis onto the empty stack, adding a right parenthesis at the end of INEXP, and initializing POSTEXP with null. The algorithm terminates when stack become empty.
The algorithm contains following steps

1. Initialize POSTEXP with null
2. Add ')' at the end of INEXP
3. Create an empty stack and push '(' on to the stack
4. Initialize i=0,j=0
5. Do while stack is not empty
6. If INEXP[i] is an operand then
   POSTEXP[j]=INEXP[i]
   I=i+1
   j=j+1
   Goto step 5
7. If INEXP[i] is '(' then
   push (INEXP[i])
   i=i+1
   Goto step 5
8. If INEXP[i] is an operator then
   While precedence of the operator at the top of the stack > precedence of operator
   POSTEXP[j]=pop()
   J=j+1
   End of while
   Push (INEXP[i])
   I=i+1
   Goto step 5
9. If INEXP[i] is ')' then
   While the operator at the top of the stack is not '('
   POSTEXP[j]=pop()
   J=j+1
   End while
   Pop()
10. End of step 5
11. End algorithm

For example convert the infix expression (A+B)*(C-D)/E into postfix expression showing stack status after every step.

Symbol scanned from infix	Stack status (bold letter shows the top of the stack)	Postfix expression
	(	
(	((**	
A	((**	A
+	((**+**	A
B	((**+**	AB
)	**(**	AB+
*	(****	AB+
(	(*(**	AB+

C	(***(***	AB+C
-	(*(**-**	AB+C
D	(*(**-**	AB+CD
)	(*****	AB+CD-
/	(***/***	AB+CD-*
E	(***/***	AB+CD-*E
)		AB+CD-*E/

Answer: Postfix expression of **(A+B)*(C-D)/E** is **AB+CD-*E/**

Evaluation of Postfix expression using Stack

Algorithm to evaluate a postfix expression P.

1. Create an empty stack
2. i=0
3. while P[i] != NULL
   - if P[i] is operand then
     - Push(P[i])
     - I=i+1
   - Else if P[i] is a operator then
     - Operand2=pop()
     - Operand1=pop()
     - Push (Operand1 operator Operator2)
   - End if
4. End of while
5. return pop() // return the calculated value which available in the stack.

   End of algorithm

Example: Evaluate the following postfix expression showing stack status after every step

8, 2, +, 5, 3, -, *, 4 /

token scanned from postfix expression	Stack status (bold letter shows the top of the stack) after processing the scanned token	Operation performed
8	**8**	Push 8
2	8, **2**	Push 2
+	**10**	Op2=pop() i.e 2 Op1=pop() i.e 8 Push(op1+op2) i.e. 8+2
5	10, **5**	Push(5)
3	10, 5, **3**	Push(3)
-	10, **2**	Op2=pop() i.e. 3 Op1=pop() i.e. 5 Push(op1-op2) i.e. 5-3
*	**20**	Op2=pop() i.e. 2 Op1=pop() i.e. 10 Push(op1-op2) i.e. 10*2
4	20, **4**	Push 4
/	5	Op2=pop() i.e. 4 Op1=pop() i.e. 20 Push(op1/op2) i.e. 20/4
NULL	Final result 5	Pop 5 and return 5

Example:

Evaluate the following Boolean postfix expression showing stack status after every step
True, False, True, AND, OR, False, NOT, AND

token scanned from postfix expression	Stack status (bold letter shows the top of the stack) after processing the scanned token	Operation performed
True	**True**	Push True
False	True, **False**	Push False
True	True, False, **True**	Push True
AND	True, **False**	Op2=pop() i.e. True Op1=pop() i.e. False Push(Op2 AND Op1) i.e. False AND True=False
OR	True	Op2=pop() i.e. False Op1=pop() i.e. True Push(Op2 OR Op1) i.e. True OR False=True
False	True, **False**	Push False
NOT	True, **True**	Op1=pop() i.e. False Push(NOT False) i.e. NOT False=True
AND	**True**	Op2=pop() i.e. True Op1=pop() i.e. True Push(Op2 AND Op1) i.e. True AND True=True
NULL	Final result **True**	Pop True and Return True

Queue

Queue is a linear data structure which follows First In First Out (FIFO) rule in which a new item is added at the rear end and deletion of item is from the front end of the queue. In a FIFO data structure, the first element added to the queue will be the first one to be removed.

Linear Queue implementation using Array

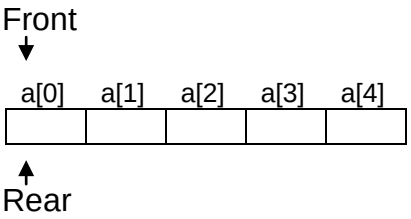
```
#include<iostream.h>
const int size=5;
class queue
{int front , rear;
int a[size];
public:
queue(){frot=0;rear=0;} //Constructor to create an empty queue
void addQ()
{ if(rear==size)
        cout<<"queue is full<<endl;
 else
        a[rear++]=item;
}
int delQ()
{if(front==rear)
        {cout<<"queue is empty"<<endl; return 0;}

else
        return a[front++];
}
}
```

```
void main()
{queue q1;
q1.addQ(3);
q1.addQ(5) ;
q1.addQ(7) ;
cout<<q1.delQ()<<endl ;
cout<<q1.delQ()<<endl ;
cout<<q1.delQ()<<endl;
cout<<q1.delQ()<<endl;
}
```

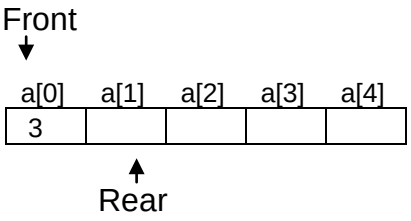
Output is

3
5
7
Queue is empty
0

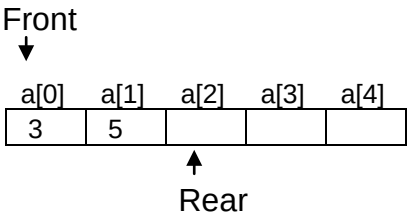
In the above program the statement **queue q1** creates an empty queue q1.



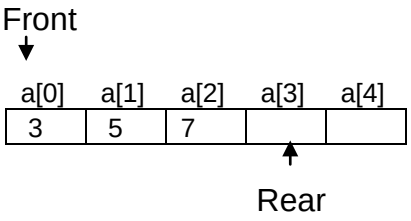
After execution of the statement **q1.addQ(3),** status of queue is



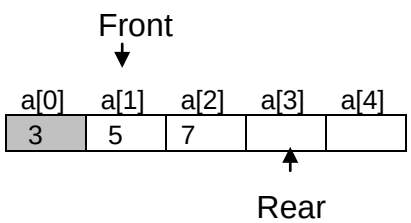
After execution of the statement **q1.addQ(5),** status of queue is



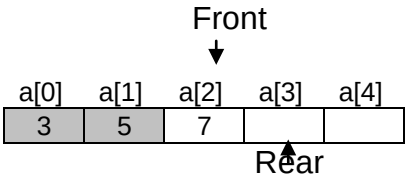
After execution of the statement **q1.addQ(7),** status of queue is



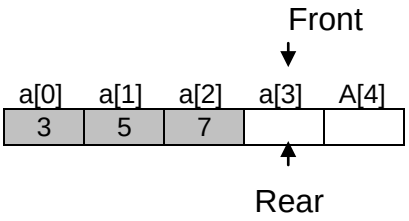
After execution of the first cout<<**q1.delQ()** statement**,** 3 is deleted from queue status of queue is



After execution of the second cout<<**q1.delQ()** statement**, 5** is deleted from the queue status of queue is



After execution of the third cout<<**q1.delQ()** statement**, 7** is deleted from the queue. The status of queue is empty



After execution of the fourth cout<<**q1.delQ()** statement**,** the message "queue is empty" displayed and status of queue is

Note that since rear and front moves only in one direction therefore once the rear cross the last element of the array(i.e. rear==size) then even after deleting some element of queue the free spaces available in queue cannot be allocated again and the function delQ() display error message "queue is full".

Queue using linked list

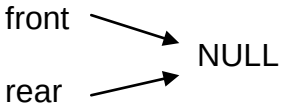
```
#include<iostream.h>
struct node
{
int item;
node *next;
};
class queue
{
node *front, *rear;
public:
queue() {front=NULL; rear=NULL;}//constructor to create empty queue
void addQ(int item);
int delQ();
};
void queue::addQ(int item)
{node * t=new node;
if(t==NULL)
        cout<<"memory not available, queue is full"<<endl;
else
        {t->item=item;
         t->next=NULL;
        if (rear==NULL)   //if the queue is empty
                {rear=t; front=t;     //rear and front both will point to the first node
                }
        else
                {
```

```
                rear->next=t;
                 rear=t;
                }
        }
}
int queue::delQ()
{
if(front==NULL)
        cout<<"queue is empty"<<return 0;
else
        {node *t=front;
         int r=t->item;
         front=front->next; //move front to the next node of the queue
         if(front==NULL)
                rear==NULL;
         delete t;
         return r;
        }
}
void main()
{
queue q1;
q1.addQ(3);
q1.addQ(5) ;
q1.addQ(7) ;
cout<<q1.delQ()<<endl ;
cout<<q1.delQ()<<endl ;
cout<<q1.delQ()<<endl;
cout<<q1.delQ()<<endl;
}
```

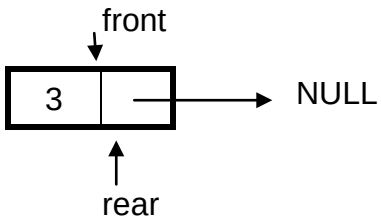
Output is

3
5
7
Queue is empty 0

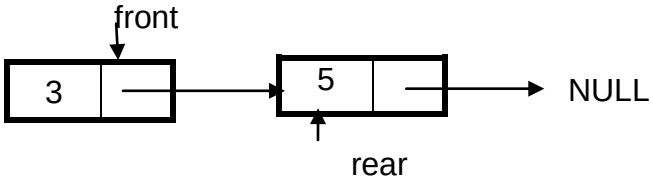
In the above program the statement **queue q1;** invokes the constructor queue() which create an empty queue object q1 and initialize front and rear with NULL.



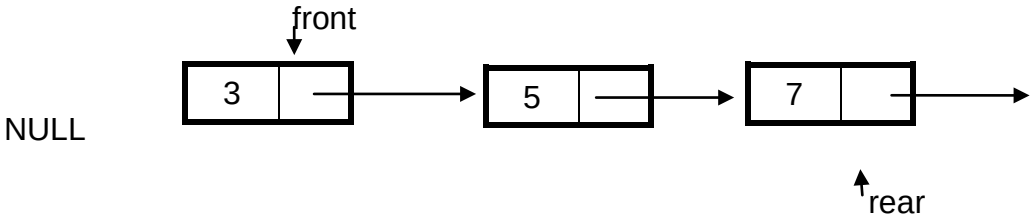
After statement q1.addQ(3) the stack become



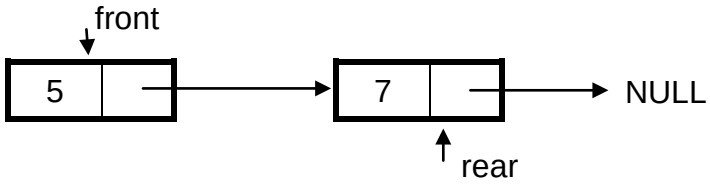
After statement q1.addQ(5) the stack become



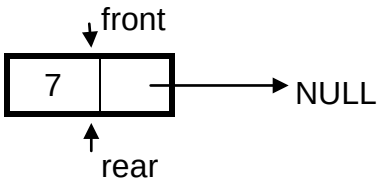
After statement q1.addQ(7) the stack become



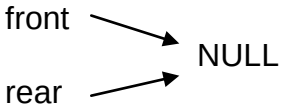
After the first q1.delQ() statement the node currently pointed by front (i.e. node containing 3) is deleted from the queue, after deletion the status of queue is



After the second q1.delQ() statement the node currently pointed by front (i.e. node containing 5) is deleted from the queue, after deletion the status of queue is



After the third q1.delQ() statement the node currently pointed by front (i.e. node containing 7) is deleted from the queue, after deletion the queue become empty therefore NULL is a assigned to both rear and front



After the fourth q1.delQ() statement, the error message “queue is empty“ displayed and the pop() function return 0 to indicate that queue is empty.

Circular queue using array

Circular queues overcome the problem of unutilised space in linear queues implemented as array. In circular queue using array the rear and front moves in a circle from 0,1,2…size-1,0, and so on.

```
#include<iostream.h>
const int size=4;
class Cqueue
{
int a[size];
int front,rear,anyitem;
public:
void Cqueue(){front=0;rear=0;anyitem=0;}
void addCQ(int item);
int delCQ();
};
void Cqueue::addCQ(int item)
{
If(front==rear && anyitem>0)
        cout<<”Cqueue is full”<<endl;
else
        {a[rear]=item;
```

```
        rear=(rear+1)%size; //rear will move in circular order
        anyitem++;   //value of the anyitem contains no of items present in the queue
        }
}
int Cqueue::delCQ()
{
if(front==rear&& anyitem==0)
        cout<<"Cqueue is empty"<<endl; return 0; //0 indicate that Cqueue is empty
else
        {int r=a[front];
        front=(front+1)/size;
        anyitem--;
        }
}

void main()
{Cqueue q1;
q1.addCQ(3);
q1.addCQ(5) ;
cout<<q1.delCQ()<<endl ;
q1.addCQ(7) ;
cout<<q1.delCQ()<<endl ;
q1.addCQ(8) ;
q1.addCQ(9) ;
cout<<q1.delCQ()<<endl;
cout<<q1.delCQ()<<endl;
}
```

Output is

3
5
7
8

Stack, Queues using Arrays and Linked List : Practice Questions

2 Marks Questions

1. Convert the following infix expressions to postfix expressions using stack
    1. A + (B * C) ^ D – (E / F – G)
    2. A * B / C * D ^ E * G / H
    3. ((A*B)-((C_D)*E/F)*G
    4. A+B/C*D+F*G
    5. A+B-A/(B*(A-B-A*D)^B)
    6. (B+(C+D)*(E+F)/G)/H
    7. (TRUE || FALSE) && ! (FALSE || TRUE)
    8. (A / B + C) / D + E / (F + G * H / I)
    9. (((b-(c*d-e)+f))/g)+(h*j+x)
    10. A+(((B*C)*(D+E)+F*G)^(H-J)
    11. (A-B) *(C/(D-E)+F-G
2. Evaluate the following postfix expression E given below; show the contents of the stack during the evaluation
    1. E= 5,9,+2,/,4,1,1,3,_,*,+
    2. E= 80,35,20,-,25,5,+,-,*
    3. E= 30,5,2,^,12,6,/,+,-
    4. E=15, 3, 2, +, /, 7, + 2, *
    5. E=25, 8, 3, - , / 6, *, 10 +

6. E=8, 7, -, 9, 8, *, -, 2, / , 3,4, * 2, / -
7. E= 5,20,15,-,*,25,2,*,-
8. E= 10,+,15,*,25,5,/,2+
9. E= 7,6,2,/,+,18,-
10. E= 7,6, +, 8, * , 2, -, 3, * , 2, 4, * , -
11. E=TRUE, FALSE, NOT, TRUE, OR, FALSE, AND, OR
12. E=FALSE, TRUE, TRUE, NOT, AND, OR, AND

3 or 4 Marks Questions

1. An array A[40][10] is stored in the memory along the column with each element occupying 4 bytes. Find out the address of the location A[3][6] if the location A[30][10] is stored at the address 9000.

2. An array A[30][20] is stored in the memory along the row with each element occupying 2 bytes. Find out the address of the location A[10][5] if the location A[20][10] is stored at the address 10000.

3. Define functions in C++ to perform a PUSH and POP operation in a dynamically allocated stack considering the following :

```
struct Node
{ int X,Y;
Node *Link; };
class STACK
{ Node * Top;
public:
STACK( )
 {
TOP=NULL;
}
void PUSH( );
void POP( );
~STACK( );
};
```

4. Write a function in C++ to perform a Add and Delete operation in a dynamically allocated Queue considering the following:

```
struct node
{
int empno ;char name[20] ;float sal ;
 Node *Link;
};
```

DATABASE AND SQL

Basic Database concepts

Data :- Raw facts and figures which are useful to an organization. We cannot take decisions on the basis of data.

Information:- Well processed data is called information. We can take decisions on the basis of Information

Database:- Collection of interrelated data . It is basically a computer based record keeping system that serve multiple applications.

Advantages of Database over file system

- Data Independence can be achieved i.e. data and programs that manipulate the data are two different entities.
- The amount of data redundancy(repetition of data) in stored data can be reduced.
- Stored Data can be shared by single or multiple users.
- Data Integrity (i.e. when database contains only accurate data) can be maintained
- Data Security can be easily implemented.

Relational Data Model:

Relational data model is introduced by C.F.Codd in 1970.
The relational data model describes the world as "a collection of inter-related relations (or tables) ".
In other way ,in relational data model the data is organized into tables (i.e rows and columns) these tables are called relations.

Data redundancy:- Duplication of data is known as data redundancy

Data inconsistency:- If two entries about same data do not agree it is said to be inconsistence

Data security :- Refers to protection of data against accidental or intentional discloser to unauthentic persons.

Data independence:- It is the ability to modify a scheme in one level without affecting a scheme definition in a higher level.

Relational data model:- The data is organized into tables, these tables are called relations.

Relation :- Relation is a table that means data is organized into rows and columns.

Domain:- It is a pool of values from which the actual values present in a given column are drawn.

Tuple:- A row in a relation is called a tuple that is horizontal part of a relation and represents a record of relation

Attribute:- A column in a relation is called an attribute. It is also termed as field or data item.

Degree:- Number of attributes in a relation is called degree of a relation.

Cardinality:- Number of tuples in a relation is called cardinality of a relation.

View:- It is a virtual table that does not really exist in its own but is instead derived from one or more underlying base table.

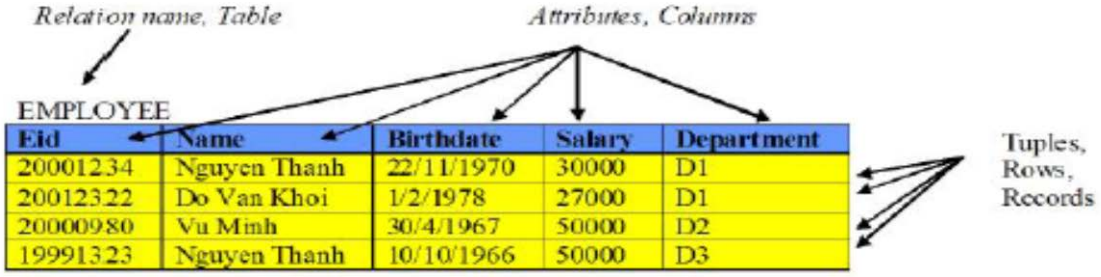
Field:- Set of characters that represents specific data element.

Record:- Collection of fields is called a record. A record can have fields of different data types.

File:- Collection of similar types of records is called a file.

Table:- Collection of rows and columns that contains useful data/information is called a table.

A table generally refers to the passive entity which is kept in secondary storage device.



Primary Key:- Primary key is set of one or more attributes that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Candidate Key: Set of all attributes which can serve as a primary key in a relation.

Alternate Key: All the candidate keys other than the primary keys of a relation are alternate keys for a relation.

Foreign Key: a non-key attribute, whose values are derived from the primary key of some other table is known as foreign key in its current table.

Employee table

Emp_id	Emp_name	salary	PAN No	Department_no	NPS_no
23008	Ramesh	50000	NALK8907P	10	32FG67
24009	kavita	35000	LMSA9845K	14	87KH86
23908	sana	45000	KMSN6789O	11	78PO98

Department table

Dept_no	Dept_name	Head
10	Marketing	Mrs. Sumitra
11	Sales	Mr. Raja raman
12	Purchasing	Mr. K P Singh
13	Manufacturing	Mr. Jyoti Ranjan
14	Store	Miss Suman

Eg:- candidate key in Employee table:- Emp_id, PAN No and NPS_no
Any one of candidate can be selected as primary key. If Emp_id is selected as Primary key then PAN No and NPS_no is alternate key
Department_no in Employee table can have only values that are present in Dept_no column of Department table so it is a foreign key in Employee table.

Relational Algebra

It is a Relational Database language which provides notations for deriving information from the permanent relations in the database.
Relational Algebra uses some specialized operators to extract data from database.
The operators are :

- σ (sigma)
- π (pi)
- X(Cartesian product)
- U (union)
- ∩ (intersection)
- - (difference)

- **The SELECT Operation (σ (sigma))**

SELECT operation is an unary operation and is used to select a subset of the tuples from a relation that satisfy the selection condition.

Syntax : σ<selection condition> (R)

σ -> used to denote the SELECT operator

<selection condition> -> is a boolean expression.

e.g.

Retrieve from the STUDENT relation the tuples for those students having level >= 2

Result : σLEVEL>=2 (STUDENT)

- **The PROJECT Operation (Π (pi))**

It is a unary operator which projects out a vertical subset of a given relation

Syntax : Π<attribute list> (R)

Query : Retrieve the room numbers of lecturers

Result : ΠROOM# (LECTURER)

- **UNION (U):** The set of tuples that are in relation R or in relation S, or in both is denoted by R U S or (R union S)

Roll	Name	Marks
11	Mounika	65
14	Mousumi	89

Relation R

Roll	Name	Marks
12	Pallavi	94
14	Mousumi	89

Relation S

Roll	Name	Marks
11	Mounika	65
12	Pallavi	94
14	Mousumi	89

R U S

- INTERSECTION (∩): The set of tuples that are in R and also in S and is denoted by R∩S or (R intersect S)

Roll	Name	Marks
14	Mousumi	89

R∩S

- SET DIFFERENCE (-) : The set of tuples that are in R but not in S is denoted by (R minus S) or R-S

Roll	Name	Marks
11	Mounika	65

- CARTESIAN PRODUCT (X)

Let R and S be relations of degree n1 and n2 respectively. Then the Cartesian Product of R and S, is of degree n1+n2. It is denoted by R X S

Suppose relation R is R(A1,A2,…,An) and relation S is S(B1,B2,….,Bm) then Q=R X S is Q(A1,A2,…An,B1,B2,….Bm)

If R has k1 tuples and S has k2 tuples then R X S will have k1*k2 tuples.

Roll	Name	Marks
12	Pallavi	94
14	Mousumi	89

Relation R

Age	Class
17	XII
14	IX

Relation S

Roll	Name	Marks	Age	Class
12	Pallavi	94	17	XII
12	Pallavi	94	14	IX
14	Mousumi	89	17	XII
14	Mousumi	89	14	IX

Relation R X S

Structured Query Language

SQL is a non procedural language that is used for accessing , handling and manipulating in the databases(relations).

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. **Data Definition Language (DDL)**

   DDL contains commands that are used to create the tables, databases, indexes, views, sequences and synonyms etc.

   e.g: Create table, create view, create index, alter table etc.

2. **Data Manipulation Language (DML)**

   DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval.

   e.g Select, Insert, Delete, Update etc.

3. **View Definition:**

   DDL contains set of command to create a view of a relation.

   e.g : create view

4. **Embedded Data Manipulation Language**

   the embedded form of SQL is designed for use within general purpose programming language

5. **Authorization :-** specify access right to relations and view.

6. **Integrity :**-sql provide limited form of integrity checking.

7. **Transaction Control Language (TCL)**

TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc.

Data dictionary :- is refers to a file that contains metadata i.e. data about data.

Data types of SQL

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

1) NUMBER
2) INTEGER
3) CHAR
4) VARCHAR / VARCHAR2
5) DATE

1. NUMBER

Used to store a numeric value in a field/column. It may be decimal, integer or a real value. General syntax is

Number(n,d)

Where **n** specifies the number of digits and
d specifies the number of digits to the right of the decimal point.

e.g marks number(3) declares marks to be of type number with maximum value 999.
pct number(5,2) declares pct to be of type number of 5 digits with two digits to the right of decimal point.

2. CHAR

Used to store character type data in a column. General syntax is

Char (size)

where size represents the maximum number of characters in a column. The CHAR type data can hold at most 255 characters.

e.g name char(25) declares a data item name of type character of upto 25 size long.

3. VARCHAR/VARCHAR2

This data type is used to store variable length alphanumeric data. General syntax is

varchar(size) / varchar2(size)

where size represents the maximum number of characters in a column. The maximum allowed size in this data type is 2000 characters.

e.g address varchar(50); address is of type varchar of upto 50 characters long.

4. DATE

Date data type is used to store dates in columns. SQL supports the various date formats other that the standard DD-MON-YY.

e.g dob date; declares dob to be of type date.

5. INTEGER

It represent a number without a decimal point.

.

SQL Commands

a. CREATE TABLE Command:

Create table command is used to create a table in SQL. It is a DDL type of command. The general syntax of creating a table is

Creating Tables

The syntax for creating a table is

create table <table> (
<column 1> <data type> [not null] [unique] [<column constraint>],
. . . . . .
<column n> <data type> [not null] [unique] [<column constraint>],
[<table constraint(s)>]
);

For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by comma. Uppercase and lowercase letters makes no difference in column names, the only place where upper

and lower case letters matter are strings comparisons. A not null Constraint means that the column cannot have null value, that is a value needs to be supplied for that column. The keyword unique specifies that no two tuples can have the same attribute value for this column.

Operators in SQL:

The following are the commonly used operators in SQL

1. Arithmetic Operators +, -, *, /
2. Relational Operators =, <, >, <=, >=, <>
3. Logical Operators OR, AND, NOT

Arithmetic operators are used to perform simple arithmetic operations.
Relational Operators are used when two values are to be compared and
Logical operators are used to connect search conditions in the WHERE Clause in SQL.

Constraints:

Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play when ever we try to insert, delete or update a record in a relation.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. CHECK
5. DEFAULT

Not null ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

Unique constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

Primary key constraint means that a column can not have duplicate values and not even a null value.

e.g. Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

Check constraint limits the values that can be inserted into a column of a table.

e.g marks number(3) check(marks>=0);

The above statement declares marks to be of type number and while inserting or updating the value in marks it is ensured that its value is always greater than or equal to zero.

Default constraint is used to specify a default value to a column of a table automatically. This default value will be used when user does not enter any value for that column.

e.g balance number(5) default = 0;

```
CREATE TABLE student (
Roll_no         number(3) primary key,
Name            varchar(25) not null,
Class           varchar(10),
Marks           number(3) check(marks>0),
City            varchar(25)  );
```

Data Modifications in SQL

After a table has been created using the create table command, tuples can be inserted into the table, or tuples can be deleted or modified.

INSERT Statement

The simplest way to insert a tuple into a table is to use the insert statement
insert into <table> [(<column i, . . . , column j>)] values (<value i, . . . , value j>);

INSERT INTO student VALUES(101,'Rohan','XI',400,'Jammu');
While inserting the record it should be checked that the values passed are of same data types as the one which is specified for that particular column.

For inserting a row interactively (from keyboard) & operator can be used.
e.g INSERT INTO student VALUES(&Roll_no',’&Name’,’&Class’,’&Marks’,’&City’);
In the above command the values for all the columns are read from keyboard and inserted into the table student.

NOTE:- In SQL we can repeat or re-execute the last command typed at SQL prompt by typing “/” key and pressing enter.

Roll_no	Name	Class	Marks	City
101	Rohan	XI	400	Jammu
102	Aneeta Chopra	XII	390	Udhampur
103	Pawan Kumar	IX	298	Amritsar
104	Rohan	IX	376	Jammu
105	Sanjay	VII	240	Gurdaspur
113	Anju Mahajan	VIII	432	Pathankot

Queries:
To retrieve information from a database we can query the databases. SQL SELECT statement is used to select rows and columns from a database/relation.

SELECT Command
This command can perform selection as well as projection.
Selection: This capability of SQL can return you the tuples form a relation with all the attributes.
Projection: This is the capability of SQL to return only specific attributes in the relation.
* SELECT * FROM student; command will display all the tuples in the relation student
* SELECT * FROM student WHERE Roll_no <=102;
The above command display only those records whose Roll_no less than or equal to 102.
Select command can also display specific attributes from a relation.
* SELECT name, class FROM student;
The above command displays only name and class attributes from student table.
* SELECT count(*) AS “Total Number of Records” FROM student;
Display the total number of records with title as “Total Number of Records” i.e an alias
We can also use arithmetic operators in select statement, like
* SELECT Roll_no, name, marks+20 FROM student;
* SELECT name, (marks/500)*100 FROM student WHERE Roll_no > 103;

Eliminating Duplicate/Redundant data

DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement.
e.g. SELECT DISTINCT name FROM student;
The above command returns

Name
Rohan
Aneeta Chopra
Pawan Kumar

Selecting a specific Rows (Where)

it will select only those rows which fulfill the specific criteria.
SELECT <column name>[, <column name>]
FROM <table name>
WHERE <condition>
Eg:- SELECT empno, name From Emp WHERE sal>50000

Relational Operators
compare two values =,>,<,>=,<=,<>
Eg:- SELECT * FROM emp WHERE Dept=20

Logical operator

i) OR :-if one of both condition is true it will select row and show

ii) AND:- if both condition is true then only it will select row and show it

iii) NOT:- show details of those who does not fulfill the criteria

Conditions based on a range
SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.
e.g. SELECT Roll_no, name FROM student WHERE Roll_no BETWENN 100 AND 103;
The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

Conditions based on a list
To specify a list of values, IN operator is used. This operator select values that match any value in the given list.
e.g. SELECT * FROM student WHERE city IN ('Jammu','Amritsar','Gurdaspur');
The above command displays all those records whose city is either Jammu or Amritsar or Gurdaspur.

Conditions based on Pattern
SQL provides two wild card characters that are used while comparing the strings with LIKE operator.
a. percent(%) Matches any string
b. Underscore(_) Matches any one character
e.g SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";
displays those records where last digit of Roll_no is 3 and may have any number of characters in front.
e.g SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";
displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

ORDER BY Clause
ORDER BY clause is used to display the result of a query in a specific order(sorted order).
The sorting can be done in ascending or in descending order. It should be kept in mind that the actual data in the database is not sorted but only the results of the query are displayed in sorted order.
e.g. SELECT name, city FROM student ORDER BY name;
The above query returns name and city columns of table student sorted by name in increasing/ascending order.
e.g. SELECT * FROM student ORDER BY city DESC;
It displays all the records of table student ordered by city in descending order.
Note:- If order is not specifies that by default the sorting will be performed in ascending order.

GROUP BY Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

```
SELECT column1, column2, ... column_n, aggregate_function (expression)
FROM tables
WHERE conditions
GROUP BY column1, column2, ... column_n;
```

aggregate_function can be a function such as SUM, COUNT, MAX, MIN, AVG etc.

```
e.g     SELECT name, COUNT(*) as "Number of employees"
        FROM student
        GROUP BY city;
```

HAVING Clause

The HAVING clause is used in combination with the GROUP BY clause. It can be used in a SELECT statement to filter the records that a GROUP BY returns.

The syntax for the HAVING clause is:

```
SELECT column1, column2, ... column_n, aggregate_function (expression)
FROM tables
WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```

```
e.g     SELECT SUM(marks) as "Total marks"
        FROM student
        GROUP BY department
        HAVING SUM(sales) > 1000;
```

Note: select statement can contain only those attribute which are already present in the group by clause.

Functions available in SQL

SQL Functions:

- **Sum**: to find sum total value of a given attribute.

select **SUM**(sal) "total" from emp;

- **Max** : Returns the maximum value of the expression

select **MAX**(sal) "Maximum" from emp;

- **Min**: Returns the Minimum value of the expression

select **MIN**(hiredate) "Minimum" from emp;

- **Avg** : returns average value of a group of records for one attribute

select **avg**(sal) "Average" from emp;

- **Count** : Returns the number of rows in the query

select **count**(*) "Count" from emp;

JOINS :

Join is a mechanism which will fetch data from multiple tables.There are 4 types of join.

1.**Equi Join** :-Which will fetch data from multiple tables when a common attribute is available among the tables.

E.X 1:-Select ename, deptno, loc from EMP, DEPT where EMP.deptno= DEPT.deptno;

2.**Non-Equi Join**:- Which will fetch data from multiple tables when no common attribute is available among the tables.

Select ename, deptno, sal,grade from EMP, SALGRADE where EMP.sal between SALGRADE .losal and SALGRADE .hisal ;

SQL provide large collection of inbuilt functions also called library functions that can be used directly in SQL statements.

1. Mathematical functions
2. String functions
3. Date & Time functions

1. Mathematical functions

Some of the commonly used mathematical functions are sum() avg(), count(), min(), max() etc.

e.g. SELECT sum(marks) FROM student;

displays the sum of all the marks in the table student.

e.g. SELECT min(Roll_no), max(marks) FROM student;

displays smallest Roll_no and highest marks in the table student.

2. String functions

These functions are used to deal with the string type values like
ASCII, LOWEWR, UPPER, LEN, LEFT, RIGHT, TRIM, LTRIM, RTRIM etc.

ASCII : Returns the ASCII code value of a character(leftmost character of string).

Syntax: ASCII(character)

SELECT ASCII('a') returns 97
SELECT ASCII('A') returns 65
SELECT ASCII('1') returns 49
SELECT ASCII('ABC') returns 65

For Upper character 'A' to 'Z' ASCII value 65 to 90
For Lower character 'A' to 'Z' ASCII value 97 to 122
For digit '0' to '9' ASCII value 48 to 57

NOTE: If no table name is specified then SQL uses Dual table which is a dummy table used for performing operations.

LOWER : Convert character strings data into lowercase.

Syntax: LOWER(string)

SELECT LOWER('STRING FUNCTION') returns string function

UPPER : Convert character strings data into Uppercase.

Syntax: UPPER(string)

SELECT UPPER('string function') returns STRING FUNCTION

LEN : Returns the length of the character string.

Syntax: LEN(string)

SELECT LEN('STRING FUNCTION') returns 15

REPLACE : Replaces all occurrences of the second string(string2) in the first string(string1) with a third string(string3).

Syntax: REPLACE('string1','string2','string3')

SELECT REPLACE('STRING FUNCTION','STRING','SQL') returns SQL Function

Returns NULL if any one of the arguments is NULL.

LEFT : Returns left part of a string with the specified number of characters counting from left.LEFT function is used to retrieve portions of the string.

Syntax: LEFT(string,integer)

SELECT LEFT('STRING FUNCTION', 6) returns STRING

RIGHT : Returns right part of a string with the specified number of characters counting from right.RIGHT function is used to retrieve portions of the string.

Syntax: RIGHT(string,integer)

SELECT RIGHT('STRING FUNCTION', 8) returns FUNCTION

LTRIM : Returns a string after removing leading blanks on Left side.(Remove left side space or blanks)

Syntax: LTRIM(string)

SELECT LTRIM(' STRING FUNCTION') returns STRING FUNCTION

RTRIM : Returns a string after removing leading blanks on Right side.(Remove right side space or blanks)
Syntax: RTRIM(string)
SELECT RTRIM('STRING FUNCTION ') returns STRING FUNCTION
REVERSE : Returns reverse of a input string.
Syntax: REVERSE(string)
SELECT REVERSE('STRING FUNCTION') returns NOITCNUF GNIRTS
REPLICATE : Repeats a input string for a specified number of times.
Syntax: REPLICATE (string, integer)
SELECT REPLICATE('FUNCTION', 3) returns FUNCTIONFUNCTIONFUNCTION
SPACE : Returns a string of repeated spaces. The SPACE function is an equivalent of using REPLICATE function to repeat spaces.
Syntax: SPACE (integer) (If integer is negative, a null string is returned.)
SELECT ('STRING') + SPACE(1) + ('FUNCTION') returns STRING FUNCTION
SUBSTRING : Returns part of a given string.
SUBSTRING function retrieves a portion of the given string starting at the specified character(startindex) to the number of characters specified(length).
Syntax: SUBSTRING (string,startindex,length)
SELECT SUBSTRING('STRING FUNCTION', 1, 6) returns STRING
SELECT SUBSTRING('STRING FUNCTION', 8, 8) returns FUNCTION

DELETE Command
To delete the record fro a table SQL provides a delete statement. General syntax is:-

DELETE FROM <table_name> [WHERE <condition>];
e.g. DELETE FROM student WHERE city = 'Jammu';
This command deletes all those records whose city is Jammu.
NOTE: It should be kept in mind that while comparing with the string type values lowercase and uppercase letters are treated as different. That is 'Jammu' and 'jammu' is different while comparing.

UPDATE Command
To update the data stored in the data base, UOPDATE command is used.

e. g. UPDATE student SET marks = marks + 100;
Increase marks of all the students by 100.

e. g. UPDATE student SET City = 'Udhampur' WHERE city = 'Jammu';
changes the city of those students to Udhampur whose city is Jammu.

We can also update multiple columns with update command, like
e. g. UPDATE student set marks = marks + 20, city = 'Jalandhar'
WHERE city NOT IN ('Jammu','Udhampur');

CREATE VIEW Command

In SQL we can create a view of the already existing table that contains specific attributes of the table.
e. g. the table student that we created contains following fields:
Student (Roll_no, Name, Marks, Class, City)
Suppose we need to create a view **v_student** that contains Roll_no,name and class of student table, then Create View command can be used:
CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student;
The above command create a virtual table (view) named v_student that has three attributes as mentioned and all the rows under those attributes as in student table.

We can also create a view from an existing table based on some specific conditions, like CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student WHERE City <>'Jammu';

The main difference between a Table and view is that

A **Table** is a repository of data. The table resides physically in the database.
A **View** is not a part of the database's physical representation. It is created on a table or another view. It is precompiled, so that data retrieval behaves faster, and also provides a secure accessibility mechanism.

ALTER TABLE Command

In SQL if we ever need to change the structure of the database then ALTER TABLE command is used. By using this command we can add a column in the existing table, delete a column from a table or modify columns in a table.

Adding a column
The syntax to add a column is:-

ALTER TABLE table_name
ADD column_name datatype;

e.g ALTER TABLE student ADD(Address varchar(30));
The above command add a column Address to the table atudent.
If we give command
SELECT * FROM student;
The following data gets displayed on screen:

Roll_no	Name	Class	Marks	City	Address
101	Rohan	XI	400	Jammu	
102	Aneeta Chopra	XII	390	Udhampur	
103	Pawan Kumar	IX	298	Amritsar	
104	Rohan	IX	376	Jammu	
105	Sanjay	VII	240	Gurdaspur	
113	Anju MAhajan	VIII	432	Pathankot	

Note that we have just added a column and there will be no data under this attribute. UPDATE command can be used to supply values / data to this column.

Removing a column

ALTER TABLE table_name
DROP COLUMN column_name;

e.g ALTER TABLE Student
DROP COLUMN Address;
The column Address will be removed from the table student.

DROP TABLE Command

Sometimes you may need to drop a table which is not in use. DROP TABLE command is used to
Delete / drop a table permanently. It should be kept in mind that we can not drop a table if it contains
records. That is first all the rows of the table have to be deleted and only then the table can be
dropped. The general syntax of this command is:-

DROP TABLE <table_name>;

e.g DROP TABLE student;

This command will remove the table student

Questions : Database and SQL : 1 mark questions

Q1 Write SQL queries to perform the following based on the table PRODUCT having fields as (prod_id, prod_name, quantity, unit_rate, price, city)

i. Display those records from table PRODUCT where prod_id is more than 100.
ii. List records from table PRODUCT where prod_name is 'Almirah'
iii. List all those records whose price is between 200 and 500.
iv. Display the product names whose price is less than the average of price.
v. Show the total number of records in the table PRODUCT.

Q2. Define the terms:

i. Database Abstraction
ii. Data inconsistency
iii. Conceptual level of database implementation/abstraction
iv. Primary Key
v. Candidate Key
vi. Relational Algebra
vii Domain

6 Marks Questions SQL

Q1 Write SQL commands for (i) to (viii) on the basis of relations given below:

BOOKS

book_id	Book_name	author_name	Publishers	Price	Type	qty
k0001	Let us C	Sanjay mukharjee	EPB	450	Comp	15
p0001	Genuine	J. Mukhi	FIRST PUBL.	755	Fiction	24
m0001	Mastering c++	Kanetkar	EPB	165	Comp	60
n0002	Vc++ advance	P. Purohit	TDH	250	Comp	45
k0002	Near to heart	Sanjeev	FIRST PUBL.	350	Fiction	30

ISSUED

Book_ID	Qty_Issued
L02	13
L04	5
L05	21

i. To show the books of FIRST PUBL Publishers written by P.Purohit.

ii. To display cost of all the books written for FIRST PUBL.
iii. Depreciate the price of all books of EPB publishers by 5%.
iv. To display the BOOK_NAME,price of the books whose more than 3 copies have been issued.
v. To show total cost of books of each type.
vi. To show the detail of the most costly book.

Q2. Write SQL commands for (a) to (f) and write output for (g) on the basis of PRODUCTS relation given below:

PRODUCT TABLE

PCODE	PNAME	COMPANY	PRICE	STOCK	MANUFACTURE	WARRANTY
P001	TV	BPL	10000	200	12-JAN-2008	3
P002	TV	SONY	12000	150	23-MAR-2007	4
P003	PC	LENOVO	39000	100	09-APR-2008	2
P004	PC	COMPAQ	38000	120	20-JUN-2009	2
P005	HANDYCAM	SONY	18000	250	23-MAR-2007	3

a) To show details of all PCs with stock more than 110.
b) To list the company which gives warranty for more than 2 years.
c) To find stock value of the BPL company where stock value is sum of the products of price and stock.
d) To show number of products from each company.
e) To count the number of PRODUCTS which shall be out of warranty on 20-NOV-2010.
f) To show the PRODUCT name which are within warranty as on date.
g). Give the output of following statement.
(i) Select COUNT(distinct company) from PRODUCT.
(ii) Select MAX(price)from PRODUCT where WARRANTY<=3

Answers: 1 mark questions

Q1
Ans i: **select * from product where prod_id > 100;**
Ans ii: select * from product where prod_name = 'Almirah';
Ans iii: select * from product where price between 200 and 500;
Ans iv: select prod_name from product where price < avg(price);
Ans v: select count(*) from product;

Q2. Define the terms:

i. Database Abstraction
Ans: Database system provides the users only that much information that is required by them, and hides certain details like, how the data is stored and maintained in database at hardware level. This concept/process is Database abstraction.

ii. Data inconsistency
Ans: When two or more entries about the same data do not agree i.e. when one of them stores the updated information and the other does not, it results in data inconsistency in the database.

iii. Conceptual level of database implementation/abstraction
Ans: It describes what data are actually stored in the database. It also describes the relationships existing among data. At this level the database is described logically in terms of simple data-structures.

iv. Primary Key
Ans : It is a key/attribute or a set of attributes that can uniquely identify tuples within the relation.

v. Candidate Key
Ans : All attributes combinations inside a relation that can serve as primary key are candidate key as they are candidates for being as a primary key or a part of it.

vi. Relational Algebra

Ans : It is the collections of rules and operations on relations(tables). The various operations are selection, projection, Cartesian product, union, set difference and intersection, and joining of relations.

vii. Domain

Ans : it is the pool or collection of data from which the actual values appearing in a given column are drawn.

Answers: 6 Marks Questions:

Q1.

Ans i: select * from books where publishers='FIRST PUBL'
Ans ii: select sum(price*qty) from books where publishers='FIRST PUBL';
Ansiii: update books set price=price-0.5*price where publishers='EPB';
Ans iv: select BOOK_NAME,price from books, issued where
books.book_id=issued.book_id and quantity_issued>3;
Ans v: select sum(price*qty) from books group by type;
Ans vi: select * from books where price=(select max(price) from books));

Q2.

Ans a: select * from products where pname='TV' and stock>110;
Ans b: select company from products where warranty>2;
Ans c: select sum(price*stock) from PRODUCTS where company='BPL';
Ans d: select company,COUNT(*) from products group by company;
Ans e: select count(*) from products where ('20-NOV-2010'- manufacture)/365>warranty;
Ans f: select pname from products where (sysdate- manufacture)/365<warranty;

Ansg (i): 4
Ans(ii): 39000

Some practice questions from Database and SQL :

2 marks questions

1. What is relation? What is the difference between a tuple and an attribute?
2. Define the following terminologies used in Relational Algebra:
   (i) selection (ii) projection (iii) union (iv) Cartesian product
3. What are DDL and DML?
4. Differentiate between primary key and candidate key in a relation?
5. What do you understand by the terms **Cardinality** and **Degree** of a relation in relational database?
6. Differentiate between DDL and DML. Mention the 2 commands for each caterogy.

6 marks questions

1.

Table : SchoolBus

Rtno	Area_overed	Capacity	Noofstudents	Distance	Transporter	Charges
1	Vasant kunj	100	120	10	Shivamtravels	100000
2	Hauz Khas	80	80	10	Anand travels	85000
3	Pitampura	60	55	30	Anand travels	60000
4	Rohini	100	90	35	Anand travels	100000
5	Yamuna Vihar	50	60	20	Bhalla Co.	55000
6	Krishna Nagar	70	80	30	Yadav Co.	80000
7	Vasundhara	100	110	20	Yadav Co.	100000
8	Paschim Vihar	40	40	20	Speed travels	55000

9	Saket	120	120	10	Speed travels	100000
10	Jank Puri	100	100	20	Kisan Tours	95000

(b) To show all information of students where capacity is more than the no of student in order of rtno.

(c) To show area_covered for buses covering more than 20 km., but charges less then 80000.

(d) To show transporter wise total no. of students traveling.

(e) To show rtno, area_covered and average cost per student for all routes where average cost per student is - charges/noofstudents.

(f) Add a new record with following data:
(11, " Moti bagh",35,32,10," kisan tours ", 35000)

(g) Give the output considering the original relation as given:
(i) select sum(distance) from schoolbus where transporter= " Yadav travels";
(ii) select min(noofstudents) from schoolbus;
(iii) select avg(charges) from schoolbus where transporter= " Anand travels";
(v) select distinct transporter from schoolbus;

2.

TABLE : GRADUATE

S.NO	NAME	STIPEND	SUBJECT	AVERAGE	DIV.
1	KARAN	400	PHYSICS	68	I
2	DIWAKAR	450	COMP. Sc.	68	I
3	DIVYA	300	CHEMISTRY	62	I
4	REKHA	350	PHYSICS	63	I
5	ARJUN	500	MATHS	70	I
6	SABINA	400	CEHMISTRY	55	II
7	JOHN	250	PHYSICS	64	I
8	ROBERT	450	MATHS	68	I
9	RUBINA	500	COMP. Sc.	62	I
10	VIKAS	400	MATHS	57	II

(a) List the names of those students who have obtained DIV 1 sorted by NAME.

(b) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.

(c) To count the number of students who are either PHYSICS or COMPUTER SC graduates.

(d) To insert a new row in the GRADUATE table: 11,"KAJOL", 300, "computer sc", 75, 1

(e) Give the output of following sql statement based on table GRADUATE:
(i) Select MIN(AVERAGE) from GRADUATE where SUBJECT="PHYSICS";
(ii) Select SUM(STIPEND) from GRADUATE WHERE div=2;
(iii) Select AVG(STIPEND) from GRADUATE where AVERAGE>=65;
(iv) Select COUNT(distinct SUBDJECT) from GRADUATE;

(f) Assume that there is one more table GUIDE in the database as shown below:

Table: GUIDE

MAINAREA	ADVISOR
PHYSICS	VINOD
COMPUTER SC	ALOK
CHEMISTRY	RAJAN
MATHEMATICS	MAHESH

g) What will be the output of the following query:
SELECT NAME, ADVISOR FROM GRADUATE,GUIDE WHERE SUBJECT= MAINAREA;

3. Write SQL command for (i) to (vii) on the basis of the table SPORTS

Table: SPORTS

Student NO	Class	Name	Game1	Grade	Game2	Grade2
10	7	Sammer	Cricket	B	Swimming	A
11	8	Sujit	Tennis	A	Skating	C
12	7	Kamal	Swimming	B	Football	B
13	7	Venna	Tennis	C	Tennis	A
14	9	Archana	Basketball	A	Cricket	A
15	10	Arpit	Cricket	A	Atheletics	C

(a) Display the names of the students who have grade 'C' in either Game1 or Game2 or both.
(b) Display the number of students getting grade 'A' in Cricket.
(c) Display the names of the students who have same game for both Game1 and Game2.
(d) Display the games taken up by the students, whose name starts with 'A'.
(e) Assign a value 200 for Marks for all those who are getting grade 'B' or grade 'A' in both Game1 and Game2.
(f) Arrange the whole table in the alphabetical order of Name.
(g) Add a new column named 'Marks'.

4.Write SQL command for (i) to (vii) on the basis of the table Employees & EmpSalary

Table: Employees

Empid	**Firstname**	**Lastname**	**Address**	**City**
010	Ravi	Kumar	Raj nagar	GZB
105	Harry	Waltor	Gandhi nagar	GZB
152	Sam	Tones	33 Elm St.	Paris
215	Sarah	Ackerman	440 U.S. 110	Upton
244	Manila	Sengupta	24 Friends street	New Delhi
300	Robert	Samuel	9 Fifth Cross	Washington
335	Ritu	Tondon	Shastri Nagar	GZB
400	Rachel	Lee	121 Harrison St.	New York
441	Peter	Thompson	11 Red Road	Paris

Table: EmpSalary

Empid	**Salary**	**Benefits**	**Designation**
010	75000	15000	Manager
105	65000	15000	Manager
152	80000	25000	Director
215	75000	12500	Manager
244	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
400	32000	7500	Salesman
441	28000	7500	salesman

Write the **SQL commands** for the following :
(i) To show firstname,lastname,address and city of all employees living in paris

(ii) To display the content of Employees table in descending order of Firstname.

(iii) To display the firstname,lastname and total salary of all managers from the tables Employee and empsalary , where total salary is calculated as salary+benefits.

(iv) To display the maximum salary among managers and clerks from the table Empsalary.

Give the **Output** of following SQL commands:

(i) Select firstname,salary from employees ,empsalary where designation = 'Salesman' and Employees.empid=Empsalary.empid;

(ii) Select count(distinct designation) from empsalary;

(iii) Select designation, sum(salary) from empsalary group by designation having count(*) >2;

(iv) Select sum(benefits) from empsalary where designation ='Clerk';

Boolean Algebra

Boolean algebra is an algebra that deals with Boolean values((TRUE and FALSE) . In daily life we
normally asks questions like should I go for shopping or not? Should I watch TV or not? etc.
The answers to these questions will be either yes or no, true or false, 1 or 0, which are truth values.

Truth table:
Truth table is a table, which represents all the possible values of logical variables/statements along with all the possible results of given combinations of values.

Logical Operators:
Logical operators are derived from the Boolean algebra, which is the mathematical representation of the concepts without going into the meaning of the concepts.
1. NOT Operator—Operates on single variable. It gives the complement value of variable.

X	$\overline{\mathbf{X}}$
0	1
1	0

2. OR Operator -It is a binary operator and denotes logical Addition operation and is represented by "+" symbol

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

3. **AND Operator** – AND Operator performs logical multiplications and symbol is (.) dot.

0.0=0
0.1=0
1.0=0
1.1=1

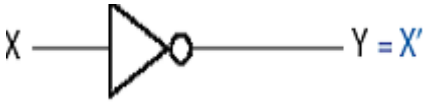
Truth table:

X	Y	X.Y
0	0	0
0	1	0
1	0	0
1	1	1

Basic Logic Gates
A gate is simply an electronic circuit, which operates on one or more signals to produce an output signal. Gates are digital circuits because the input and output signals are either low (0) or high (1). Gates also called logic circuits.
There are three types of logic gates:

1. Inverter (NOT gate)
2. OR gate
3. AND gate

1. NOT gate : This gate takes one input and gives a single output. The symbol of this logic gate is



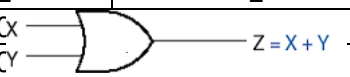
This circuit is used to obtain the compliment of a value.
If X = 0, then X' = 1.
The truth table for NOT gate is :

X	$\overline{X}$
0	1
1	0

2. OR gate : The OR gate has two or more input signals but only one output signal if any of the input signal is 1(high) the output signal is 1(high).

Truth Table for Two Input OR gate is :

X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1



X	Y	Z	F=X+Y+Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

The circuit diagram of two inputs OR gate is:-

AND gate The AND gate have two or more than two input signals and produce an output signal. When all the inputs are 1(High) then the output is 1 otherwise output is 0 only.

X	Y	F=X.Y

0	0	0
0	1	0
1	0	0
1	1	1

Circuit diagram of Two input AND gate

Basic postulates of Boolean Algebra:

Boolean algebra consists of fundamental laws that are based on theorem of Boolean algebra. These fundamental laws are known as basic postulates of Boolean algebra. These postulates states basic relations in boolean algebra, that follow:

I If X != 0 then x=1 and If X!=1 then x=0

II OR relations(logical addition)

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1

III AND relations (logical multiplication)

0.0=0
0.1=0
1.0=0
1.1=1

IV Complement Rules

$\overline{0} = 1, \qquad 1 = \overline{0}$

Principal of Duality

This principal states that we can derive a Boolean relation from another Boolean relation by performing simple steps. The steps are:-

1. Change each AND(.) with an OR(+) sign
2. Change each OR(+) with an AND(.) sign
3. Replace each 0 with 1 and each 1 with 0

e.g

0+0=0 then dual is 1.1=1
1+0=1 then dual is 0.1=0

Basic theorem of Boolean algebra

Basic postulates of Boolean algebra are used to define basic theorems of Boolean algebra that provides all the tools necessary for manipulating Boolean expression.

1. Properties of 0 and 1
   (a) 0+X=X
   (b) 1+X=1
   (c) 0.X=0
   (d) 1.X=X
2. Indempotence Law
   (a) X+X=X
   (b) X.X=X
3. Involution Law

$\overline{\overline{(X)}} = X$

4. Complementarity Law

(a) $X + \overline{X}=1$

(b) $X.\overline{X}=0$

5. Commutative Law

(a) $X+Y=Y+X$

(b) $X.Y=Y.X$

6. Associative Law

(a) $X+(Y+Z)=(X+Y)+Z$

(b) $X(YZ)=(XY)Z$

7. Distributive Law

(a) $X(Y+Z)=XY\_XZ$

(b) $X=YZ=(X+Y)(X+Z)$

8. Absorption Law

(a) $X+XY= X$

(b) $X(X+Y)=X$

Some other rules of Boolean algebra

$X+X\overline{Y}=X+Y$

Demorgan's Theorem

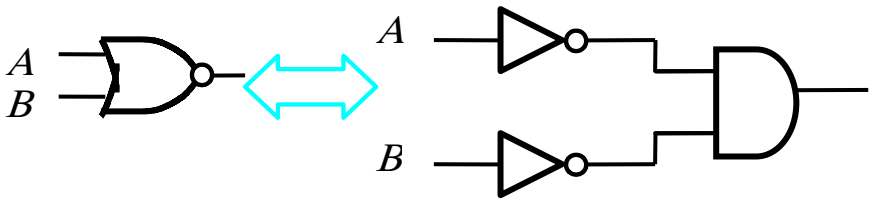
A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra.

Demorgan's First Theorem:

This rule states that the compliment of OR of two operands is same as the AND of the compliments of those operands.

Mathematically it can be written as:-

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

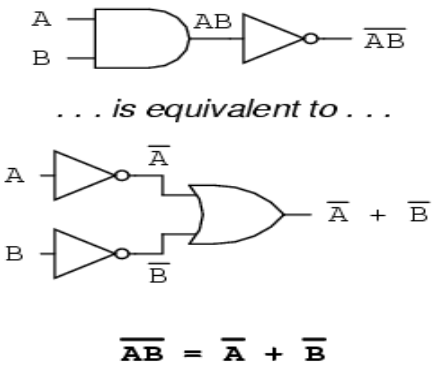


Demorgan's Second Theorem:

This rule states that the compliment of AND of two operands is same as the OR of the compliments of those operands.

Mathematically it can be written as:-

$$\overline{A \cdot B} = \overline{A} + \overline{B}$$



$$\overline{AB} = \overline{A} + \overline{B}$$

Derivation of Boolean expression:-

Minterms and Maxterms

- Consider two binary variables *x* and *y* combined with an AND operation.
  x'y', x'y, xy', xy

Each of these four AND terms represents one of the distinct areas in the Venn diagram and is called a *minterm* or *standard product*.

- Consider two binary variables *x* and *y* combined with an OR operation.
  x'+ y', x'+ y, x + y', x + y

Each of these four OR terms represents one of the distinct areas in the Venn diagram and is called a *maxterm* or *standard sum*.

- *n* Variables can be combined to form 2^n minterms or maxterms.

Minterms and Maxterms for Three Binary Variables						
			Minterms		**Maxterms**	
x	**Y**	**Z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	x'y'z'	m_0	x+y+z	M_0
0	0	1	x'y'z	m_1	x+y+z'	M_1
0	1	0	x'yz'	m_2	x+y'+z	M_2
0	1	1	x'yz	m_3	x+y'+z'	M_3
1	0	0	xy'z'	m_4	x'+y+z	M_4
1	0	1	xy'z	m_5	x'+y+z'	M_5
1	1	0	xyz'	m_6	x'+y'+z	M_6
1	1	1	xyz	m_7	x'+y'+z'	M_7

- A Boolean function may be represented algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Functions of Three Variables				
X	***y***	***z***	**Function *F1***	**Function *F2***
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

F1= x'y'z + xy'z' + xyz = $m_1 + m_4 + m_7$
F2= x'yz + xy'z + xyz' + xyz = $m_3 + m_5 + m_6 + m_7$

- The complement of a Boolean function may be read from the truth table by forming a minterm for each combination that produces a 0 in the function and then ORing those terms.
  $F1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$

Example: Express the Boolean function $F(A,B,C) = AB + C$ as a sum of minterms.

Step 1 – Each term must contain all variables

$AB = AB(C + C') = ABC + ABC'$

$C = C(A + A') = AC + A'C$

$= AC(B + B') + A'C(B + B')$

$= ABC + AB'C + A'BC + A'B'C$

Step 2 – OR all new terms, eliminating duplicates

$F(A,B,C) = A'B'C + A'BC + AB'C + ABC' + ABC$

$= m_1 + m_3 + m_5 + m_6 + m_7$

$= \Sigma(1, 3, 5, 6, 7)$

Example: Express the Boolean function $F(x,y,z) = x'y + xz$ as a product of maxterms.

Step 1 – Convert the function into OR terms using the distributive law

$F(x,y,z) = (x'y + x)(x'y + z)$

$= (x + x')(y + x)(x' + z)(y + z)$

$= (y + x)(x' + z)(y + z)$

Step 2 – Each term must contain all variables

$y + x = y + x + zz' = (x + y + z)(x + y + z')$

$x' + z = x' + z + yy' = (x' + y + z)(x' + y' + z)$

$y + z = y + z + xx' = (x + y + z)(x' + y + z)$

step 3 – AND all new terms, eliminating duplicates

$F(x,y,z) = (x + y + z)(x + y + z')(x' + y + z)(x' + y' + z)$

$= (M_0 M_1 M_4 M_6)$

$= \Pi(0, 1, 4, 6)$

Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms that make the function equal to 1, whereas its complement is a 1 for those minterms that the function is 0.

Example : $F(A,B,C) = \Sigma(0, 2, 4, 6, 7)$

$F'(A,B,C) = \Sigma(1, 3, 5) = m_1 + m_3 + m_5$

Take the complement of F' by DeMorgan's theorem to obtain F in a different form:

$F(A,B,C) = (m_1 + m_3 + m_5)' = (m_1' \cdot m_3' \cdot m_5') = M_1M_3M_5 = \Pi(1, 3, 5)$

- To convert from one canonical form to the other, interchange the symbols Σ and Π, and list those numbers missing from the original form.

Standard Forms

- The two canonical forms of Boolean algebra are basic forms that one obtains from reading a function from the truth table. By definition, each minterm or maxterm must contain all variables in either complemented or uncomplemented form.
- Another way to express Boolean functions is in *standard form*. In this configuration, the terms that form the function may contain one, two, or any number of literals.
- There are two types of standard forms: the *sum of products* and the *product of sums*.
- The sum of products is a Boolean function containing AND terms, called *product terms*, of one or more literals each. The sum denotes the ORing of these terms.
  Example: $F1 = y' + xy + x'yz'$

- The product of sums is a Boolean expression containing OR terms, called *sum terms*. Each term may have one or more literals. The product denotes the ANDing of these terms.
  Example: *F2 = x(y' + z)(x' + y + z' + w)*
- A Boolean function may also be expressed in nonstandard form.
  Example: *F3 = (AB + CD)(A'B' + C'D')*

Minimization of Boolean expressions:-
After obtaining SOP and POS expressions, the next step is to simplify the Boolean expression.
There are two methods of simplification of Boolean expressions.

1. **Algebraic Method**
2. **Karnaugh Map :**

1.Algebric method: This method makes use of Boolean postulates, rules and theorems to simplify the expression.

Example..1. Simplify ABCD + ABCD +ABCD +ABCD

Solution-- $A\bar{B}C\bar{D} + A\bar{B}CD + ABC\bar{D} + ABCD$

$= A\bar{B}C(\bar{D}+D) + ABC(\bar{D}+D)$

$= A\bar{B}C.1 + ABC.1$ $(\bar{D}+D=1)$

$= AC(\bar{B}+B)$
$= AC.1 = AC$

Example..2.. Reduce.. $X`Y`Z`+\bar{X}`\bar{Y}\bar{Z}`+\bar{X}Y`\bar{Z}`+X\bar{Y}\bar{Z}`$

Solution… $\bar{X}\,\bar{Y}\,\bar{Z}+\bar{X}Y\bar{Z}+X\bar{Y}\bar{Z}+XYZ$

$= \bar{X}\,(\bar{Y}\,\bar{Z}+Y\bar{Z}) + X(\bar{Y}\,\bar{Z}+Y\bar{Z})$

$= \bar{X}\,(\bar{Z}\,(\bar{Y}+Y)) + X(Z(\bar{Y}+Y))$

$= \bar{X}(\bar{Z}.1) + X(\bar{Z}.1)$ $(\bar{Y}+Y=1)$

$= \bar{X}\bar{Z}+X\bar{Z}$

$= \bar{Z}(\bar{X}+X)$

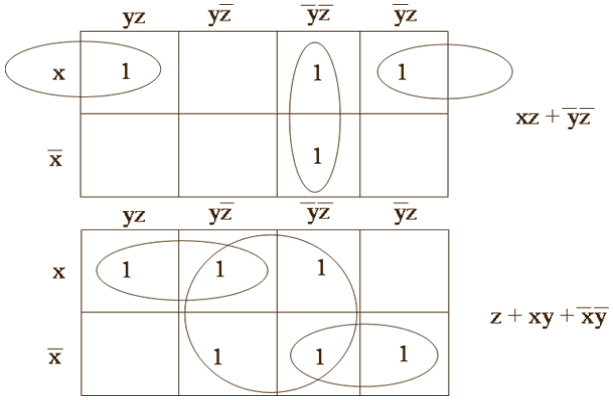
$= \bar{Z}.1$

$= \bar{Z}$

2. Using Karnaugh Map :
A Karnaugh map is graphical display of the fundamental products in a truth table.
For example:

- Put a 1 in the box for any minterm that appears in the SOP expansion.
- Basic idea is to cover the **largest adjacent** blocks you can whose side length is some power of 2.
- Blocks can "wrap around" the edges.
- For example, the first K-map here represents $xy + x\bar{y} = x(y + \bar{y}) = x$. (since y+y'=1)

- The second K-map, similarly, shows $xy + \overline{x}y = (x + \overline{x})y = y$



- Remember, group together adjacent cells of 1s, to form largest possible rectangles of sizes that are powers of 2.
- Notice that you can overlap the blocks if necessary.

Sum Of Products Reduction using K- Map

- In SOP reduction each square represent a minterm.
- Suppose the given function is f(A,B,C,D) = ∑ (0,2,7,8,10,15)
- Enter the 1 in the corresponding position for each minterm of the given function.
- Now the K-map will be as follows

AB \ **CD**	C'D'	C'D	CD	CD'
A'B'	1 0	1	3	1 2
A'B	4	5	1 7	6
AB	12	13	1 15	14
A'B	1 8	9	11	1 10

For reducing the expression first mark Octet, Quad, Pair then single.

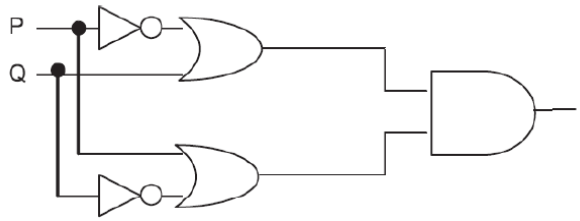
- Pair: Two adjacent 1's makes a pair.
- Quad: Four adjacent 1's makes a quad.
- Octet: Eight adjacent 1's makes an Octet.
- Pair removes one variable.
- Quad removes two variables.
- Octet removes three variables.

Reduction of expression: When moving vertically or horizontally in pair or a quad or an octet it can be observed that only one variable gets changed that can be eliminated directly in the expression.
For Example
In the above Ex
Step 1 : In K Map while moving from m_7 to m_{15} the variable A is changing its state Hence it can be removed directly, the solution becomes **B.CD = BCD.** This can be continued for all the pairs, Quads, and Octets.

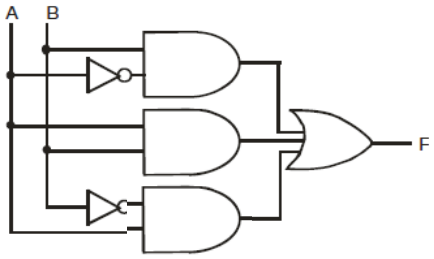
Step 2 : In K map while moving from m_0 to m_8 and m_2 to m_{10} the variable A is changing its state. Hence **B'** can be taken similarly while moving from m_0 to m_2 and m_8 to m_{10} the variable C is changing its state. Hence **D'** can be taken; the solution becomes **B'.D'**
The solution for above expression using K map is BCD + B'D'.

2 Marks Questions

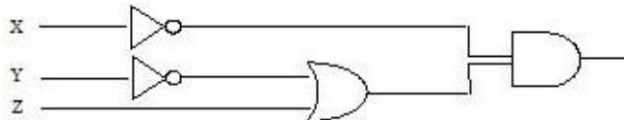
1. Write the equivalent Boolean Expression for the following Logic Circuit



2. Write the equivalent Boolean Expression F for the following circuit diagram :



3. Write the equivalent Boolean Expression F for the following circuit diagram :



4. Convert the following Boolean expression into its equivalent Canonical Sum of Product Form((SOP)

(X'+Y+Z').(X'+Y+Z).(X'+Y'+Z).(X'+Y'+Z')

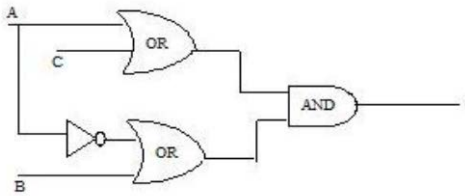
5. Convert the following Boolean expression into its equivalent Canonical Product of Sum form (POS):

A.B'.C + A'.B.C +A'.B.C'

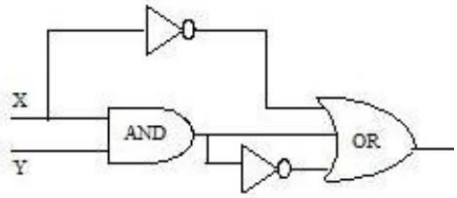
6. Draw a Logical Circuit Diagram for the following Boolean expression:

A.(B+C')

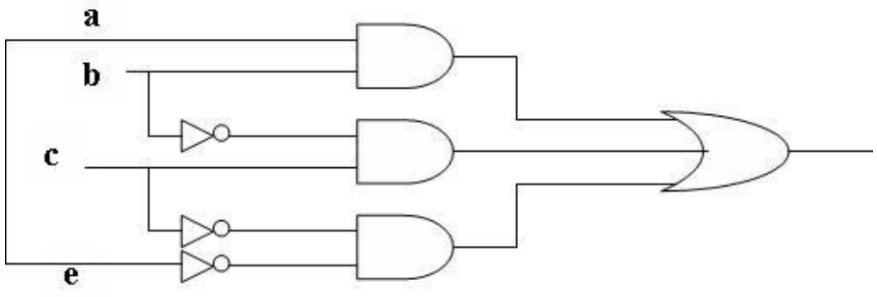
7. Write the equivalent Boolean Expression F for the following circuit diagram:



8. Prove that XY+YZ+YZ'=Y algebraically
9. Express the F(X,Z)=X+X'Z into canonical SOP form.
10. Write the equivalent Boolean Expression for the following Logic Circuit.



11. Interpret the following logical circuit as Boolean expression



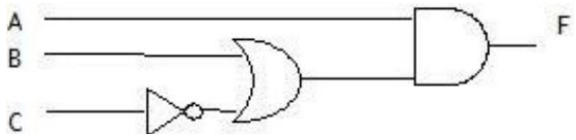
12. Design (A+B).(C+D) using NAND Gate
13. Prove x'.y'+y.z = x'yz+x'yz'+xyz+x'yz algebraically.
14. Prove that (a'+b')(a'+b)(a+b')=a'b'.
15. A Boolean function F defined on three input variable X,Y,Z is 1 if and only if the number of 1(One) input is odd (e.g. F is 1 if X=1,Y=0,Z=0). Draw the truth table for the above function and express it in canonical sum of product form.

3 Marks Questions : Boolean Algebra

1. If F(a,b,c,d)=∑(0,2,4,5,7,8,10,12,13,15), obtain the simplified form using K-Map.
2. If F(a,b,c,d) =∑(0,3,4,5,7,8,9,11,12,13,15), obtain the simplified form using KMap
3. Obtain a simplified form for a boolean expression
F(U,V,W,Z)= π (0,1,3,5,6,7,10,14,15)
4. Reduce the following boolean expression using K-Map
F(A,B,C,D) = ∑(5,6,7,8,9,12,13,14,15)

Answers: 2 Marks Questions : Boolean Algebra

1. F(P,Q)=(P'+Q).(P+Q')
2. A'B+AB+AB'
3. X'(Y'+Z)
4. F(X, Y, Z) =π(4 , 5 , 6 , 7)
=∑(0, 1 , 2 , 3)
= X'. Y'. Z' + X'. Y'. Z + X'. Y. Z' + X'. Y. Z
5. A.B'.C + A'.B.C +A'.B.C' = π (0,1,4,6,7)
OR=(A+B+C).(A+B+C').(A'+B+C).(A'+B'+C).(A'+B'+C')
6.



7. (A+C)(A'+B)
8. XY+YZ+YZ'=Y

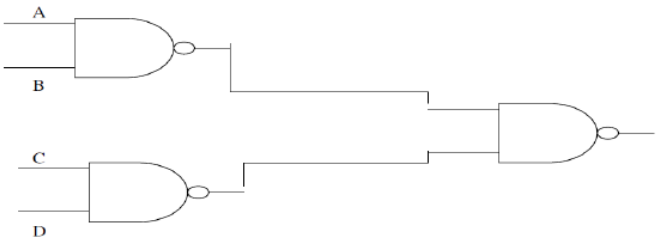
L.H.S.
XY+YZ+YZ'
= XY+Y(Z+Z')
=XY+Y=Y(X+1)
=Y.1
=Y=RHS

9. F(X,Z)=X+X'Z =X(Y+Y')+X'(Y+Y')Z
=XY+XY'+X'YZ+X'Y'Z
=XY(Z+Z')+XY'(Z+Z')+X'YZ+X'Y'Z
=XYZ+XYZ'+XY'Z+XY'Z'+X'YZ+X'Y'Z

10. XY+(XY)'+X'

11. ab+b'c+c'e'

12.



13. L.H.S.= x'y +y.z
=x'y.1+1.y.z =x'y(z+z')+(x+x')y.z
=x'yz+x'yz'+xyz+x'yz =RHS

14. LHS=(a'+b')(a'+b)(a+b')
=(a'a'+a'b+a'b'+b'b)(a+b')
=(a'+a'b+a'b'+0)(a+b')
=aa'+a'b'+aa'b+a'bb'+a'ab'+a'b'b'
=0+a'b'+0+)+0+0+a'b'=a'b'=RHS

15.

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

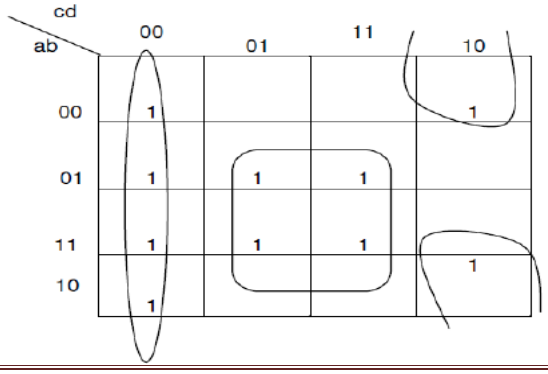
Canonical SOP
XYZ'+XY'Z+XY'Z'+XYZ

Answers: 3 marks Questions
Boolean Algebra

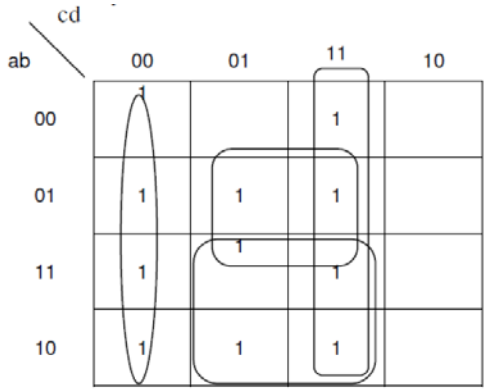
1. F(a,b,c,d)=∑(0,2,4,5,7,8,10,12,13,15)

F(a,b,c,d)=B1+B2+B3
B1=m0+m4+m12+m8==c'd'
B2=m5+m7+m13+m15=bd
B3=m0+m2+m8+m10=b'd'

F(a,b,c,d)=c'd'+bd+b'd'

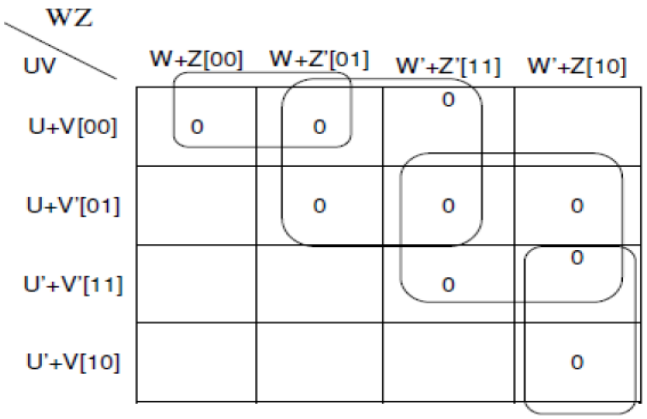


2. F(a,b,c,d) =∑(0,3,4,5,7,8,9,11,12,13,15)



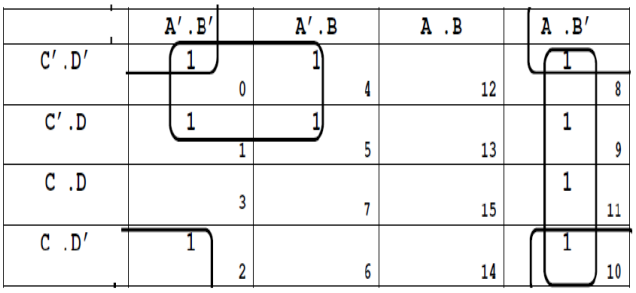
F(a,b,c,d)=B1+B2+B3+B4
B1=m0+m4+m12+m8=c'd'
B2=m5+m7+m13+m15=bd
B3=m13+m15+m9+m11=ad
B4=m3+m7+m15+m11=cd
F(a,b,c,d)=c'd'+bd+ad+cd

3. F(U,V,W,Z)= π (0,1,3,5,6,7,10,14,15)



F(U,V,W,Z)=B1.B2.B3.B4
B1=M0.M1=(U+V+W)
B2=M1.M3.M5.M7=(U+Z')
B3=M7.M6.M15.M14=(V'+W')
B4=M14.M10=(U'+W'+Z)
F(U,V,W,Z)= (U+V+W).(U+Z').(V'+W').(U'+W'+Z)

4. F(A,B,C,D) = ∑(5,6,7,8,9,12,13,14,15)



F(A,B,C,D)= A'.C'+A.B' + B'.D'

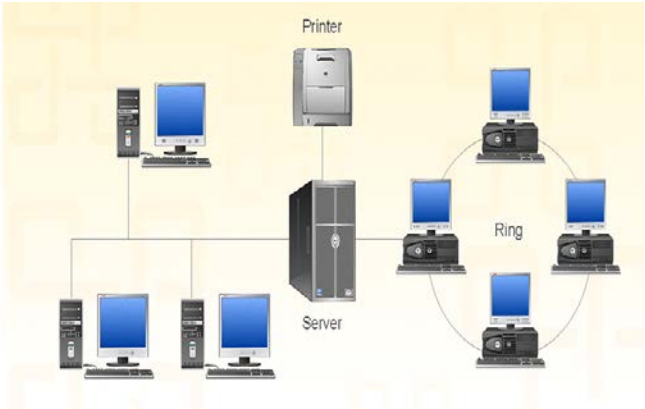
F(A,B,C,D)=B1+B2+B3
B1=M0+M2+M8+M10=B'D'
B2=M0+M1+M4+M5=A'C'
B3=M8+M9+M11+M10=AB'
F(A,B,C,D)=B'D'+A'C'+AB'

COMMUNICATION AND NETWORK CONCEPTS

Points to remember

Network

- The collection of interconnected computers is called a computer network.
- Two computers are said to be interconnected if they are capable of sharing and exchanging information.



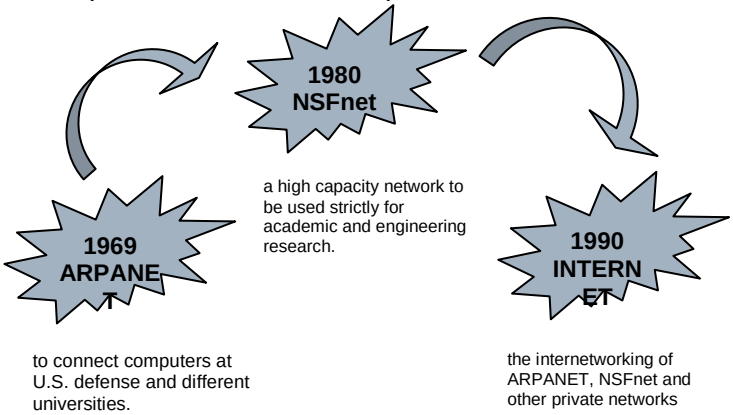
Usages of Networking:

- Resource Sharing
- Reliability
- Cost Factor
- Communication Medium

Resource Sharing means to make all programs, data and peripherals available to anyone on the network irrespective of the physical location of the resources and the user.
Reliability means to keep the copy of a file on two or more different machines, so if one of them is unavailable (due to some hardware crash or any other) them its other copy can be used.
Cost factor means it greatly reduces the cost since the resources can be shared
Communication Medium means one can send messages and whatever the changes at one end are done can be immediately noticed at another.

Evolution of Networking

1969 - First network came into existence**ARPANET** (ADVANCED RESEARCH PROJECT AGENCY NETWORK)MID 80'S - **NSFNET** (NATIONAL SCIENCE FOUNDATION



NETWORK)

- **Internet** is the network of networks.

SWITCHING TECHNIQUES

Switching techniques are used for transmitting data across networks.
Different types are :

- Circuit Switching
- Message Switching
- Packet Switching

Circuit Switching

- *Circuit switching* is the transmission technology that has been used since the first communication networks in the nineteenth century.
- First the complete physical connection between two computers is established and then the data are transmitted from the source computer to the destination.
- When a call is placed the switching equipment within the system seeks out a physical copper path all the way from the sender to the receiver.
- It is must to setup an end-to-end connection between computers before any data can be sent.
- The circuit is *terminated* when the connection is closed.
- In circuit switching, resources remain allocated during the full length of a communication, after a circuit is established and until the circuit is terminated and the allocated resources are freed.

Message Switching

- In this the source computer sends data or the message to the switching circuit which stores the data in its buffer.
- Then using any free link to the switching circuit the data is send to the switching circuit.
- Entire message is sent to the destination. It reaches through different intermediate nodes following the "store and forward" approach.
- No dedicated connection is required.

Packet Switching

- Conceived in the 1960's, *packet switching* is a more recent technology than circuit switching.
- Packet switching introduces the idea of cutting data i.e. at the source entire message is broken in smaller pieces called packets which are transmitted over a network without any resource being allocated.
- Then each packet is transmitted and each packet may follow any rout available and at destination packets may reach in random order.
- If no data is available at the sender at some point during a communication, then no packet is transmitted over the network and no resources are wasted.
- At the destination when all packets are received they are merged to form the original message.
- In packet switching all the packets of fixed size are stored in main memory.

DATA COMMUNICATION TERMINOLOGIES

Data channel	• The information / data carry from one end to another in the network by channel.

Baud & bits per second (bps)	• It's used to measurement for the information carry of a communication channel**.** • Measurement Units :- • bit • 1 Byte= 8 bits • 1 KBPS (Kilo Byte Per Second)= 1024 Bytes , • 1 Kbps (kilobits Per Second) = 1024 bits, • 1 Mbps (Mega bits Per Second)=1024 Kbps
Bandwidth	• It is amount of information transmitted or receives per unit time. • It is measuring in Kbps/Mbps etc unit**.**

Transmission Media

- data is transmitted over copper wires, fiber optic cable, radio and microwaves. the term 'media' is used to generically refer to the physical connectors, wires or devices used to plug things together.

Basic communications media types

- Copper
  - unshielded twisted pair (utp)
  - shielded twisted pair (stp)
  - coaxial cable (thinnet, thicknet)
- Fiber optic
  - single-mode
  - multi-mode
- Infrared
- Radio & microwave

Twisted Pair Cable

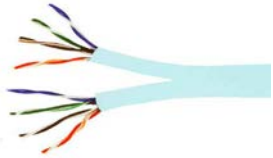
- These cables consist of two insulated copper wires twisted around each other in a double helix.
- Twisting of wires reduces crosstalk which is bleeding of a signal from one wire to another.

Types:

- Unshielded Twisted Pair (UTP)
- Shielded Twisted Pair (STP

STP offers greater protection from interference and crosstalk due to shielding.
But it is heavier and costlier than UTP.

USE 1. In local telephone communication
2. For digital data transmission over short distances upto 1 km



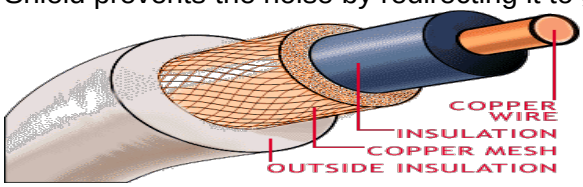
Advantages:

- Easy to install and maintain
- Simple
- Inexpensive
- Low weight
- Suitable for small (Local) Networks

Disadvantages:

- Not suitable for long distance due to high attenuation.
- Low bandwidth support.
- Low Speed

Coaxial cable

- Coaxial cable consists of a solid copper wire core surrounded by a plastic cladding shielded in a wire mesh.
- Shield prevents the noise by redirecting it to ground**.**
- 

Types:

Coaxial cable comes in two sizes which are called *thinnet* and *thicknet.*

- Thicknet : segment length upto 500 m
- Thinnet : segment length upto 185 m

USE:

In TV channel communication

Advantages:

- Better than twisted wire cable.
- Popular for TV networks.
- Offers higher bandwidth & Speed

- **Disadvantages:**

- Expensive than twisted wires.
- Not compatible with twisted wire cable.

Optical Fibres

- Thin strands of glass or glass like material designed to carry light from one source to another.
- Source converts (Modulates) the data signal into light using LED (Light Emitting Diodes) or LASER diodes and send it over the Optical fiber.

It consists of three parts:

1. The core: glass or plastic through which the light travels.
2. The cladding : covers the core and reflects light back to the core
3. Protective coating : protects the fiber

Advantages

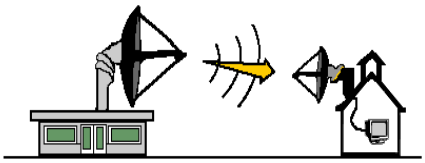
- Not affected by any kind of noise.
- High transmission capacity
- Speed of Light
- Suitable for broadband communication

Disadvantages

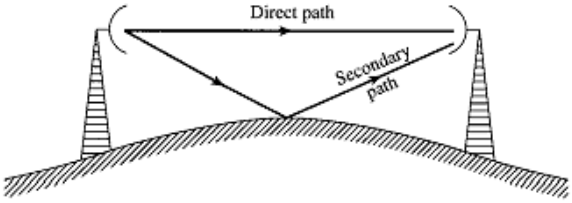
- Installation requires care.
- Connecting two Optical fibers is difficult.
- Optical fibers are more difficult to solder
- Most expensive

Microwaves

Microwaves are transmitted from the transmitters placed at very high towers to the receivers at a long distance.



Microwaves are transmitted in line of sight fashion, and also propagated through the



surfaces.

Advantages

- Maintenance easy than cables.
- Suitable when cable can not be used.

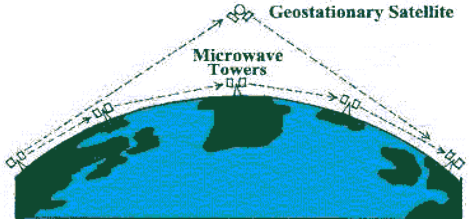
Disadvantages

- Repeaters are required for long distance communication.
- Less Bandwidth available

Satellite

Geostationary satellites are placed around 36000 KM away from the earth's surface. In satellite communication transmitting station transmits the signals to the satellite. (It is called up-linking). After receiving the signals (microwaves) it amplifies them and transmit back to earth in whole visibility area.

Receiving stations at different places can receive these signals. (It is called down-linking).



Advantage

- Area coverage is too large

Disadvantage

- High investment

Network devices

Modem

- A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.
- Modem means Modulation/ Demodulation.
- Modulation: A modem changes the digital data from your computer into analog data, a format that can be carried by telephone lines.
- Demodulation: The modem receiving the call then changes the analog signal back into digital data that the computer can digest.
- The shift of digital data into analog data and back again, allows two computers to speak with one another.

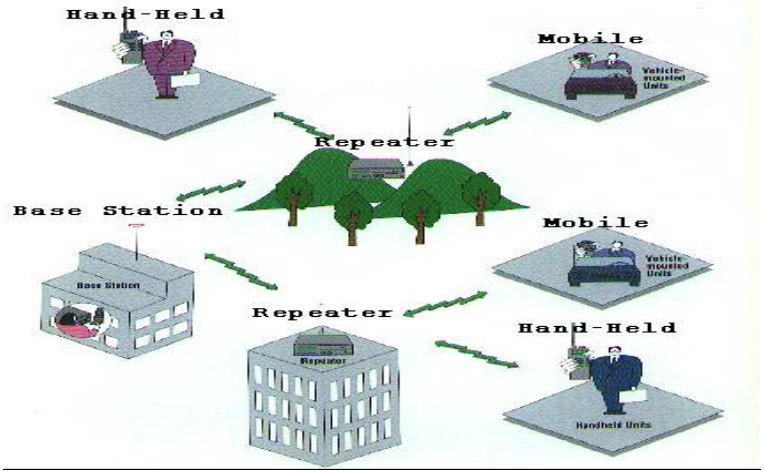
RJ- 45 Connector

RJ-45 is short for Registered Jack-45. It is an eight wire connector which is commonly used to connect computers on the local area networks i.e., LAN.

Network Interface Cards (Ethernet Card)

- A network card, network adapter or NIC (network interface card) is a piece of computer hardware designed to allow computers to communicate over a **computer network**. It provides physical access to a networking medium and often provides a low-level addressing system through the use of MAC addresses. It allows users to connect to each other either by using cables or wirelessly.

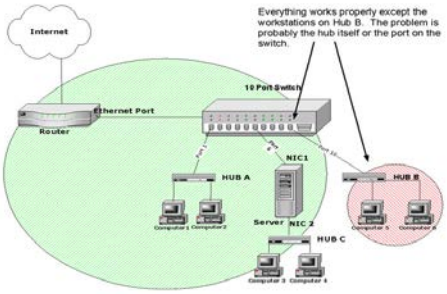
Repeaters



A repeater is an electronic device that receives a signal and retransmits it at a higher level or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances without degradation. In most twisted pair Ethernet configurations, repeaters are required for cable runs longer than 100 meters.

Hubs

A hub contains multiple ports. When a packet arrives at one port, it is copied to all the ports of the hub. When the packets are copied, the destination address in the frame does not change to a broadcast address. It does this in a rudimentary way, it simply copies the data to all of the Nodes connected to the hub.



BridgesA network bridge connects multiple network segments at the data link layer (layer 2) of the OSI model. Bridges do not promiscuously copy traffic to all ports, as hubs do, but learn which MAC addresses are reachable through specific ports. Once the bridge associates a port and an address, it will send traffic for that address only to that port. Bridges do send broadcasts to all ports except the one on which the broadcast was received.

.

Switches

Switch is a device that performs switching. Specifically, it forwards and filters OSI layer 2 datagrams (chunk of data communication) between ports (connected cables) based on the Mac-Addresses in the packets. This is distinct from a hub in that it only forwards the

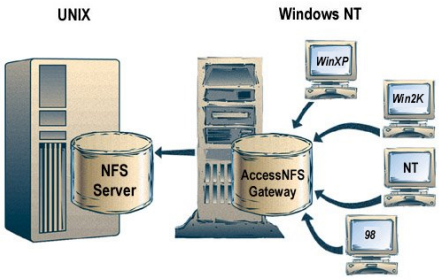
datagrams to the ports involved in the communications rather than all ports connected. Strictly speaking, a switch is not capable of routing traffic based on IP address (layer 3) which is necessary for communicating between network segments or within a large or complex LAN.

Routers

- Routers are networking devices that forward data packets between networks using headers and forwarding tables to determine the best path to forward the packets. Routers work at the network layer of the TCP/IP model or layer 3 of the OSI model. Routers also provide interconnectivity between like and unlike media (RFC 1812).
- A router is connected to at least two networks, commonly two LANs or WANs or a LAN and its ISP's network.

GATEWAY

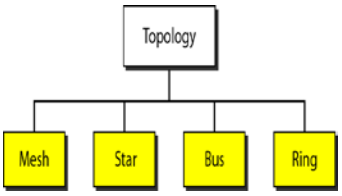
A Gateway is a network device that connects dissimilar networks. It established an intelligent connection between a local area network and external networks with completely different structures.



Network topologies and types

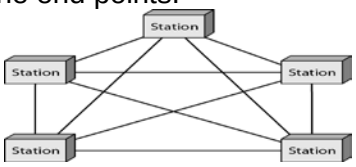
Network topology

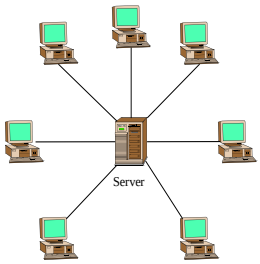
- Computer networks may be classified according to the network topology upon which the network is based, such as Bus network, Star network, Ring network, Mesh network, Star-bus network, Tree or Hierarchical topology network, etc.
- Network Topology signifies the way in which intelligent devices in the network see their logical relations to one another.



Mesh Topology

- The value of fully meshed networks is proportional to the exponent of the number of subscribers, assuming that communicating groups of any two endpoints, up to and including all the end points.





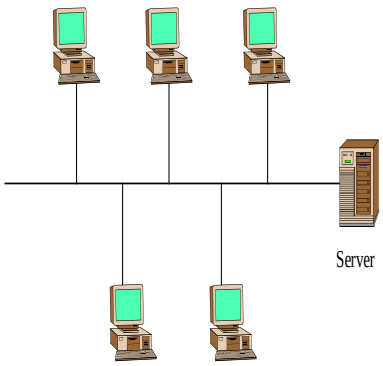
Star Topology

The type of network topology in which each of the nodes of the

network is connected to a central node with a point-to-point link in a 'hub' and 'spoke' fashion, the central node being the 'hub' and the nodes that are attached to the central node being the 'spokes' (e.g., a collection of point-to-point links from the peripheral nodes that converge at a central node) – all data that is transmitted between nodes in the network is transmitted to this central node, which is usually some type of device that then retransmits the data to some or all of the other nodes in the network, although the central node may also be a simple common connection point (such as a 'punch-down' block) without any active device to repeat the signals.

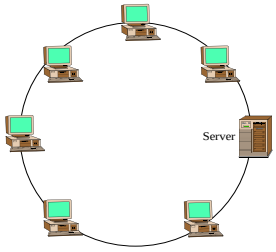
Bus Topology

- The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the 'bus', which is also commonly referred to as the backbone, or trunk) – all data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able to be received by all nodes in the network virtually simultaneously (disregarding propagation delays).



Ring Topology

- The type of network topology in which each of the nodes of the network is connected to two other nodes in the network and with the first and last nodes being connected to each other, forming a ring – all data that is transmitted between nodes in the network travels from one node to the next node in a circular manner and the data generally flows in a single direction only***.***



Compter Networks

- A communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as printers or storage devices

Local Area Network (LAN)

A network covering a small geographic area, like a home, office, or building. Current LANs are most likely to be based on Ethernet technology. For example, a library will have a wired or a communications network is two or more computers connected to share data and resources are “networked.” The simple idea behind computer networking is to allow users to access more information and give them access to devices not directly attached to their “local” system, such as **printers or storage devices.**

Metropolitan Area Network (MAN)

- A Metropolitan Area Network is a network that connects two or more Local Area Networks or Campus Area Networks together but does not extend beyond the boundaries of the immediate town, city, or metropolitan area. Multiple routers, switches & hubs are connected to create a MAN.

Wide Area Network (WAN)

- WAN is a data communications network that covers a relatively broad geographic area (i.e. one city to another and one country to another country) and that often uses transmission facilities provided by common carriers, such as telephone companies. WAN technologies generally function at the lower three layers of the OSI reference model: the physical layer, the data link layer, and the network layer.

Network Protocols

Protocols

- A protocol means the rules that are applicable for a network.
- It defines the standardized format for data packets, techniques for detecting and correcting errors and so on.
- A protocol is a formal description of message formats and the rules that two or more machines must follow to exchange those messages.
- E.g. using library books.

Types of protocols are:

1. HTTP
2. FTP
3. TCP/IP
4. SLIP/PPP

- **Hypertext Transfer Protocol** (**HTTP**) is a communications protocol for the transfer of information on the intranet and the World Wide Web. Its original purpose was to provide a way to publish and retrieve hypertext pages over the Internet.
- HTTP is a request/response standard between a client and a server. A client is the end-user; the server is the web site.

- **FTP (File Transfer Protocol)** is the simplest and most secure way to exchange files over the Internet. The objectives of FTP are:
- To promote sharing of files (computer programs and/or data).
- To encourage indirect or implicit use of remote computers.
- To shield a user from variations in file storage systems among different hosts.
- To transfer data reliably, and efficiently.

- **TCP/IP (Transmission Control Protocol / Internet Protocol)**

TCP - is responsible for verifying the correct delivery of data from client to server. Data can be lost in the intermediate network. TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.
IP - is responsible for moving packet of data from node to node. IP forwards each packet based on a four byte destination address (the IP number). The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

- **SLIP/PPP (Serial Line Internet Protocol / Point to Point Protocol)**

SLIP/PPP provides the ability to transport TCP/IP traffic ever serial line between two computers. The home user's computer has a communications link to the internet. The home user's computer has the networking software that can speak TCP/IP with other computers on the Internet. The home user's computer has an identifying address (IP address) at which it can be contacted by other computers on Internet. E.g. dial up connection.

Telnet-
It is an older internet utility that lets us log on to remote computer system. It also facilitates for terminal emulation purpose. Terminal emulation means using a pc like a mainframe computer through networking.
(i) Run telnet client- Type telnet in run dialog box.
(ii) Connect to telnet site -specify the host name, port and terminal type.
(iii) Start browsing- surf the shown site with provided instruction.
(iv) Finally disconnect-press Alt+F4.

Wireless/Mobile Computing

Wireless communication is simply data communication without the use of landlines. Mobile computing means that the computing device is not continuously connected to the base or central network.

1. **GSM(Global System for Mobile communication):** it is leading digital cellular system. In covered areas, cell phone users can buy one phone that will work any where the standard is supported. It uses narrowband TDMA, which allows eight simultaneous calls on the same radio frequency.
2. **CDMA(Code Division Multiple Access)**: it is a digital cellular technology that uses spread-spectrum techniques. CDMA does not assign a specific frequency to each user. Instead ,every channel uses the full available spectrum.
3. **WLL(Wireless in Local Loop) :** WLL is a system that connects subscribers to the public switched telephone network using radio signals as a substitute for other connecting media.
4. **Email(Electronic Mail):** Email is sending and receiving messages by computer.
5. **Chat:** Online textual talk in real time , is called Chatting.
6. **Video Conferencing**: a two way videophone conversation among multiple participants is called video conferencing.
7. **SMS(Short Message Service):** SMS is the transmission of short text messages to and from a mobile pone, fax machine and or IP address.
8. **3G and EDGE:** 3G is a specification for the third generation of mobile communication of mobile communication technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationary.

EDGE(Enhanced Data rates for Global Evolution) is a radio based high speed mobile data standard.

NETWORK SECURITY CONCEPTS

Protection methods

1 Authorization - Authorization confirms the service requestor's credentials. It determines if the service requestor is entitled to perform that operation.
2 Authentication - Each entity involved in using a web service the requestor, the provider and the broker(if there is one) - is what it actually claims to be.
3 Encryption – conversion of the form of data from one form to another form.
4 Biometric System - involves unique aspect of a person's body such as Finger-prints, retinal patterns etc to establish his/her Identity.
5 Firewall - A system designed to prevent unauthorized access to or from a private network is called firewall. it can be implemented in both hardware and software or combination or both.

There are several types of firewall techniques-

* **Packet filter-** accepts or rejects of packets based on user defined

rules.

* **Application gateway-** security mechanism to specific application like FTP and Telnet servers.

* **Circuit level gateway -** applies security mechanism when a connection is established.

* **Proxy Server -** Intercepts all messages entering and leaving the network.

Cookies - Cookies are messages that a web server transmits to a web browser so that the web server can keep track of the user's activity on a specific web site. Cookies have few parameters name, value, expiration date

Hackers and crackers -

Hackers are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.

Crackers are the malicious programmers who break into secure systems.

Cyber Law -

It is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web.

WEB SERVERS

WWW (WORLD WIDE WEB)

It is a small part of Internet. It is a kind of Application of internet.It is a set of protocols that allows us to access any document on the Net through a naming system based on URLS. Internet was mainly used for obtaining textual information. But post-WWW the internet popularity grew tremendously because of graphic intensive nature of www.

Attributes of WWW

(i) **User friendly**- www resources can be easily used with the help of browser.

(ii) **Multimedia documents**-A web page may have graphic, audio, video, and animation etc at a time.

(iii) **Hypertext and hyperlinks**-the dynamic links which can move towards another web page is hyperlink.

(iv) **Interactive** -www with its pages support and enable interactivity between users and servers.

(v) **frame**-display of more than one section on single web page.

Web server- It is a WWW server that responds to the requests made by web browers.

e.g. : Apache, IIS, PWS(Personal web server for Windows 98).

Web browser- It is a WWW client that navigates through the World Wide Web and displays web pages. E.g.: FireFox Navigator, Internet Explorer etc.

Web sites- A location on a net server where different web pages are linked together by dynamic links is called a web site. Each web site has a unique address called URL.

Web page - A document that can be viewed in a web browser and residing on a web site is a web page.

Home page- a web page that is the starting page and acts as an indexed page is home page.

Web portal - that facilitates various type of the functionality as web site. for e.g. www.yahoo.com,www.rediff.com

Domain name- An internet address which is a character based is called a Domain name. Some most common domains are com, edu, gov, mil, net, org, and co.Some domain names are location based also. For e.g. au for Australia, a for Canada, in for India etc.

URL- A URL (uniform resource locator) that specifies the distinct address for each resource on the internet.e.g.http://encycle.msn.com/getinfo/stypes.asp

Web hosting - means hosting web server application on a computer system through which electronic content on the internet is readily available to any web browser client.

HTML -
It stands for Hyper Text Markup Language that facilitates to write web document that can be interpreted by any web browser. It provide certain tags that are interpreted by the browser how to display and act with the text, graphics etc. tags are specified in <>.
For e.g.
<body bgcolor=green> it is opening tag
</body> it is closing tag.
body is the tag with bgcolor attributes.
XML (eXtensible Markup Language)
XML is a markup language for documents containing structured information. Structured information contains both content (words, pictures etc.) and some indication of what role content plays.
DHTML- It stands for Dynamic Hyper Text Markup Language. DHTML refers to Web content that changes each time it is viewed. For example, the same URL could result in a different page depending on any number of parameters, such as:
*geographic location
*time of the day
*previous pages viewed by the user
*profile of the reader
WEB SCRIPTING – The process of creating and embedding scripts in a web page is known as web-scripting.
SCRIPT: A Script is a list of commands embedded in a web page. Scripts are interpreted and executed by a certain program or scripting –engine.

Types of Scripts:
1. **Client Side Script:** Client side scripting enables interaction within a web page.
Some popular client-side scripting languages are VBScript, JavaScript, PHP(Hyper Text Preprocessor).
2. Server-Side Scripts: Server-side scripting enables the completion or carrying out a task at the server-end and then sending the result to the client –end.
Some popula server-side Scripting Languages are PHP, Perl, ASP(Active Server Pages), JSP(Java Server Pages) etc.

OPEN SOURCE TERMINOLOGIES
TERMINOLOGY & DEFINITIONS:

- **Free Software:** The S/W's is freely accessible and can be freely used changed improved copied and distributed by all and payments are needed to make for free S/W.
- **Open Source Software:** S/w whose source code is available to the customer and it can be modified and redistributed without any limitation .OSS may come free of cost but nominal charges has to pay nominal charges (Support of S/W and development of S/W).
- **FLOSS (Free Libre and Open Source Software) :** S/w which is free as well as open source S/W. (Free S/W + Open Source S/W).
- **GNU (G**NU's **N**ot **U**nix**) :** GNU project emphasize on the freedom and its objective is to create a system compatible to UNIX but not identical with it.
- **FSF (Free Software Foundation) :** FSF is a non –profit organization created for the purpose of the free s/w movement. Organization funded many s/w developers to write free software.
- **OSI (Open Source Initiative)** : Open source software organization dedicated to cause of promoting open source software it specified the criteria of OSS and its source code is not freely available.
- **W3C(World Wide Web Consortium)** : W3C is responsible for producing the software standards for World Wide Web.

- **Proprietary Software:** Proprietary Software is the s/w that is neither open nor freely available, normally the source code of the Proprietary Software is not available but further distribution and modification is possible by special permission by the supplier.
- **Freeware:** Freeware are the software freely available , which permit redistribution but not modification (and their source code is not available). Freeware is distributed in *Binary Form* (ready to run) without any licensing fees.
- **Shareware:** Software for which license fee is payable after some time limit, its source code is not available and modification to the software are not allowed.
- **Localization:** localization refers to the adaptation of language, content and design to reflect local cultural sensitivities .e.g. Software Localization: where messages that a program presents to the user need to be translated into various languages.
- **Internationalization:** Opposite of localization.

OPEN SOURCE / FREE SOFTWARE

- **Linux :** Linux is a famous computer operating system . popular Linux server set of program –LAMP(Linux, Apache, MySQL, PHP)
- **Mozilla :** Mozilla is a free internet software that includes
  - a web browser
  - an email client
  - an HTML editor
  - IRC client
- **Apache server:** Apache web server is an open source web server available for many platforms such as BSD, Linux, and Microsoft Windows etc.
  - Apache Web server is maintained by open community of developers of Apache software foundation.
- **MYSQL :** MYSQL is one of the most popular open source database system. Features of MYSQl :
  - Multithreading
  - Multi –User
  - SQl Relational Database Server
  - Works in many different platform
- **PostgreSQL :** Postgres SQL is a free software object relational database server . PostgresSQL can be downloaded from www.postgressql.org.
- **Pango :** Pango project is to provide an open source framework for the layout and rendering of internationalized text into GTK + GNOME environment.Pango using Unicode for all of its encoding ,and will eventually support output in all the worlds major languages.
- **OpenOffice :** OpenOffice is an office applications suite. It is intended to compatible and directly complete with Microsoft office.
  OOo Version 1.1 includes:
  - Writer (word processor)
  - Calc(spreadsheet)
  - Draw(graphics program)
  - etc
- **Tomcat :** Tomcat functions as a servlet container. Tomcat implements the servlet and the JavaServer Pages .Tomcat comes with the jasper compiler that complies JSPs into servlets.
- **PHP(Hypertext Preprocessor) :** PHP is a widely used open source programming language for server side application and developing web content.
- **Python: Python** is an interactive programming language originally as scripting language for Amoeba OS capable of making system calls.

1or 2 Marks Questions

1. Explain function of hub and router.
Ans:
Hub: A hub contains multiple ports. When a packet arrives at one port, it is copied to all the ports of the hub. When the packets are copied, the destination address in the frame does not change to a broadcast address. It does this in a rudimentary way, it simply copies the data to all of the Nodes connected to the hub.
2. **Router :** routers are networking devices that forward data packets between networks using headers and forwarding tables to determine the best path to forward the packets

2. Expand the following terms
(i) URL (ii) ISP (iii) DHTML (iv) CDMA:
Ans; (i) URL: Unified Resource Locator
(ii) ISP: Internet Service Provider.
(iii) DHTML: Dynamic Hyper Text Markup Language

3. Differentiate between message switching and packet switching

Ans: Message Switching – In this form of switching no physical copper path is established in advance between sender and receiver. Instead when the sender has a block of data to be sent, it is stored in first switching office, then forwarded later. Packet Switching – With message switching there is no limit on block size, in contrast packet switching places a tight upper limit on block size.

4. Write two applications of Cyber Law.
Ans: Two applications of cyber law are Digital Transaction and Activities on Internet.

5. Explain GSM.
Ans: Global system for mobile, communications is a technology that uses narrowband TDMA, which allows eight simultaneous calls on the same radio frequency. TDMA is short for Time Division Multiple Access. TDMA technology uses time division multiplexing and divides a radio frequency into time slots and then allocates these slots to multiple calls thereby supporting multiple, simultaneous data channels.

6. Write difference between coaxial and optical cable.
Ans: Coaxial cable consists of a solid wire core surrounded by one or more foil or wire shield , each separated by some kind of plastic insulator. Optical fibers consists of thin strands of glass or glass like material which are so constructed that they carry light from a source at one end of the fiber to a detector at the other end.

6. Write two advantage and disadvantage of **RING** topology.
Ans:
Advantages:
1. Short cable length.
2. No wiring closet space required.
Disadvantages:
1. Node failure causes network failure
2. difficult to diagnose faults

8. Define Open Source Software, Free Software, Freeware, and Shareware**.**
Ans:
Free Software : The S/W's is freely accessible and can be freely used changed improved copied and distributed by all and payments are needed to made for free S/W.

Open Source Software : S/w whose source code is available to the customer and it can be modified and redistributed without any limitation .OSS may come free of cost but nominal charges has to pay nominal charges (Support of S/W and development of S/W).
Freeware: Freeware are the software freely available , which permit redistribution but not modification (and their source code is not available). Freeware is distributed in *Binary Form* (ready to run) without any licensing fees.
Shareware: Software for which license fee is payable after some time limit, its source code is not available and modification to the software are not allowed.

7. What is the difference between WAN and MAN?

Ans: MAN (Metropolitan Area Network) is the network spread over a city.
WAN (Wide Area Network) spread across countries.

10. What is the purpose of using FTP?

Ans: (i)To promote sharing of files (computer programs and/or data).
(ii)To encourage indirect or implicit use of remote computers

11. What is a Modem?

Ans: A modem is a computer peripheral that allows you to connect and communicate with other computers via telephone lines.

12. How is a Hacker different from a Cracker?

Ans: Hackers are more interested in gaining knowledge about computer systems and possibly using this knowledge for playful pranks.
Crackers are the malicious programmers who break into secure systems

13. Expand the following terms with respect to Networking:
(i) Modem (ii) WLL (iii) TCP/IP (iv) FTP

Ans: (i) Modem : Modulator/Demodulator
(ii) WLL: Wireless in Local Loop
(iii) TCP/IP: Transmission Control Protocol/Internet Protocol
iv) FTP: File Transfer Protocol

14. What are Protocols?

Ans: A protocol means the rules that are applicable for a network.
It defines the standardized format for data packets, techniques for detecting and correcting errors and so on.
A protocol is a formal description of message formats and the rules that two or more machines must follow to exchange those messages.
E.g. using library books.

Types of protocols are:
1. HTTP
1. FTP
2. TCP/IP
3. SLIP/PPP

15. What is the difference between Repeater and a Bridge? 1

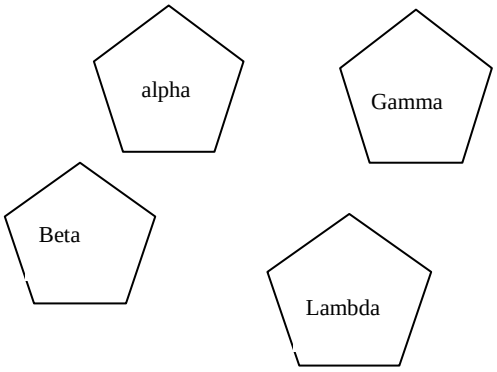
Ans: A Repeater is a network device that amplifies and restores signals for long distance transmission where as a Bridge is a network device that established an intelligent connection between two local networks with the same standard but with different types of cables.

HOTS (HIGHER ORDER THINKING SKILLS)

4 Marks Questions

1. Knowledge Supplement Organization has set up its new centre at Mangalore for its office and web based activities. It has four building as shown in the diagram below

4



Centre to Centre distance between various buildings

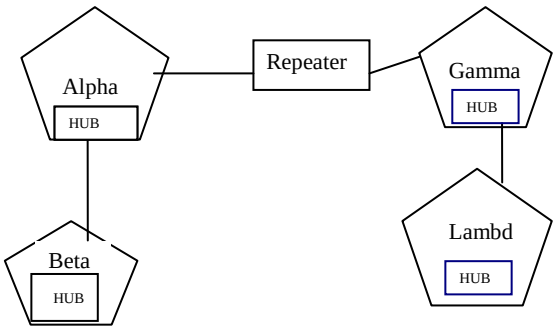
Alpha	25
Beta	50
Gamma	125
Lambda	10

Alpha to Beta	50m
Beta to gamma	150m
Gamma to Lambda	25m
Alpha to Lambda	170m
Beta to Lambda	125m
Alpha to Gamma	90m

(a) Suggesting a cable layout of connection between building state with justification where Server, Repeater and hub will be placed. 2

(b) The organization is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

Ans: (i)The most suitable place to house the server of this organization would be building Gamma , as this building contains the maximum number of computers , thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network Distance between alpha to gamma and beta to gamma is large so there repeater will require and hub is necessary for all premises because it is used in local networking.



(ii) The most economic way to connect it with a reasonable high speed would be to use radio wave transmission, as they are easy to install, can travel long distances, and penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also have the advantage of being omni directional, which is they can travel in

all the directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically.

2. Software Development Company has set up its new center at Jaipur for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below:

Center to center distances between various blocks

Black A to Block B	50 m
Block B to Block C	150 m
Block C to Block D	25 m
Block A to Block D	170 m
Block B to Block D	125 m
Block A to Block C	90 m

Number of Computers

Black A	25
Block B	50
Block C	125
Block D	10

e1) Suggest a cable layout of connections between the blocks.

e2) Suggest the most suitable place (i.e. block) to house the server of this company with a suitable reason.

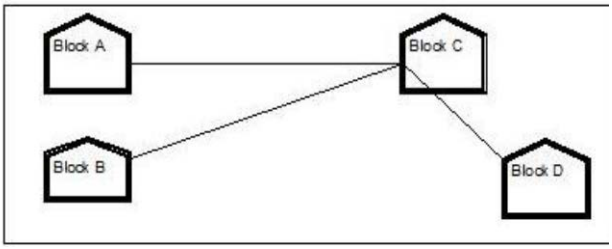
e3) Suggest the placement of the following devices with justification
(i) Repeater
(ii) Hub/Switch

e4) the company is planning to link its front office situated in the city in a hilly region where cable connection is not feasible, suggest an economic way to connect it with reasonably high speed?

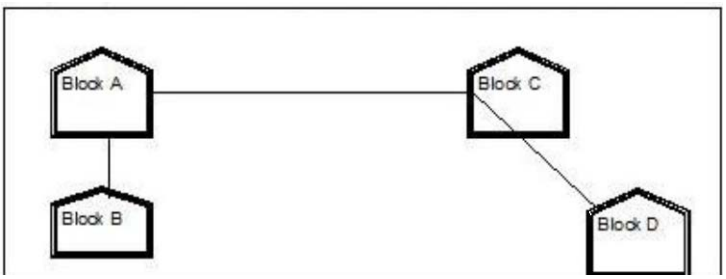
Ans:

(e1) (Any of the following option)

Layout Option 1



Layout Option 2



(e2) The most suitable place / block to house the server of this organization would be Block C, as this block contains the maximum number of computers, thus decreasing the cabling cost for most of the computers as well as increasing the efficiency of the maximum computers in the network.

(e3)

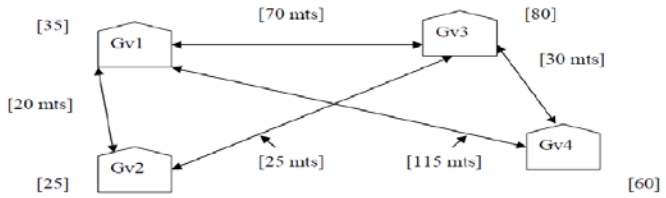
(i) For Layout 1, since the cabling distance between Blocks A and C, and that between B and C are quite large, so a repeater each would ideally be needed along their path to avoid loss of signals during the course of data flow in these routes.

For layout 2, since the distance between Blocks A and C is large so a repeater would ideally be placed in between this path.

(ii) In both the layouts, a hub/switch each would be needed in all the blocks, to Interconnect the group of cables from the different computers in each block.

(e4) The most economic way to connect it with a reasonable high speed would be to use radio wave transmission, as they are easy to install, can travel long distances, and penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also have the advantage of being omni directional, which is they can travel in all the directions from the source, so that the transmitter and receiver do not have to be carefully aligned physically.

3. Ram Goods Ltd. has following four buildings in Ahmedabad city.



[ ] – Shows computers in each building

→ - Shows distance

Computers in each building are networked but buildings are not networked so far. The company has now decided to connect building also.

(a) Suggest a cable layout for these buildings.

(b) In each of the buildings, the management wants that each LAN segment gets a dedicated bandwidth i.e. bandwidth must not be shared. How can this be achieved?

(c) The company also wants to make available shared Internet access for each of the buildings. How can this be achieved?

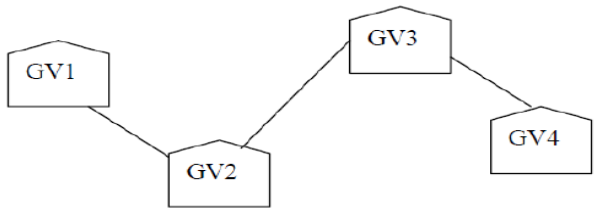
(d) The company wants to link its head office in GV1 building to its another office in Japan.

(i) Which type of transmission medium is appropriate for such a link?

(ii) What type of network would this connection result into?

Ans:

(a) Total cable length required for this layout = 75 mts



(b) To give dedicated bandwidth, the computers in each building should be connected via switches as switches offer dedicated bandwidths.

(c) By installing routers in each building, shared internet access can be made Possible.

(d) (i) Satellite as it can connect offices across globe.
(ii) WAN (Wide Area Network)

Computer Science Question Paper with Solution March 2011

Time allowed:3 hours Maximum Marks:70

1. (a) What is difference between Type Casting and Automatic Type Conversion? Also, give a suitable C++ to illustrate both.
2

Ans. Explicit Type Casting is used to convert value of one type to another type for example
float x=(float) 3 / 2; // 1.5 will be assigned as result, because 3 is converted into 3.0
Automatic Type Conversion is the type conversion done by the compiler wherever required. e. g.
float x=3/2; //here 1.0 will be assigned as result, because 1 is automatically converted in 1.0

1. (b) Write the names of the header files, which is/are essentially required to run/execute the following C++ code:
void main()
1

```
{       char CH,Text[ ]="+ve Attitude";
for(int i=0;Text[ i]='\0''i++)
        if(Text[i]==' ')
                cout<<endl;
        else    {CH=toupper(Text[i]);
                 cout<<CH;
                }
}
```

Ans. Required header files are **iostream.h** and **ctype.h**

1. (c) Rewrite the following program after removing the syntactical errors (if any). Underline each correction.
2

```
include<iostream.h>
typedef char [80] String;
 void main( )
{String S="Peace";
int L=strlen(S);
cout<<S<<' has '<<L<<'characters'<<endl;
}
```

Ans.

```
#include<iostream.h>
#include<string.h>
typedef char String[80];
void main( )
{
String S="Peace";
int L=strlen(S);
cout<<S<< "has "<<L<< "characters"<<endl;
}
```

1. (d) Find the output of the following program:
3

```
#include<iostream.h>
void SwitchOver(int A[ ], int N, int Split)
{       for (int K=0; K<N; K++)
                if(K<Split)
                        A[K]+=K;
```

```
                else
                        A[K]*=K;
}
void display(int A[ ], int N)
{       for(int K=0; K<N; K++)
                (K%2==0) ? cout<<A[K]<<"%" : cout<<A[K]<<endl;
}
void main( )
{       int H[30, 40, 50, 20, 10, 5};
        SwitchOver(H,6,3);
        Display(H, 6);
}
```

Ans.

30%41
52%60
40%25

1. (e) Find the output of the following program :

2

```
#iinclude<iostream.h>
void main( )
{       int *Queen, Moves [ ]={11,22,33,44};
        Queen=Moves;
        Moves[2]+=22;
        cout<<"Queen @"<<*Queen<<endl;
        *Queen-=11;
        Queen+=2;
        cout<<"Now @"<<*Queen<<endl;
        Queen++;
        cout<<"Finally @"<<*Queen<<endl;
        Queen++;
        cout<<"New Origin @"<<Moves[0]<<endl;
}
```

Ans.

Queen @ 11
Now @ 55
Finally @ 44
New Origin @ 0

1. (f) Go through the C++ code shown below, and find out the possible output or outputs from the suggested ouput options(i) to (iv). Also, write the minimum and maximum values, which can be assigned to the variable MyNum.

2

```
#include<iostream.h>
#include<stdlib.h>
void main( )
{       randomize();
        int MyNum,Max=5;
        MyNum=20+random(Max);
        for (int N=MyNum; N<=25; N++)
                cout<<N<<"*";
}
```

(i) **20*21*22*23*24*25**
(ii) **22*23*24*25***
(iii) **23*24***
(iv) **21*22*23*24*25**

Ans.

(ii) 22*23*24*25*

2. (a) Differentiate between Constructor and Destructor function with respect to Object Oriented Progaramming.

2

Ans. Constructor is automatically invoked when an object is created and it is used to inialize data members of the object. Constructors can be overloaded. Whereas Destructors are automatically invoked when an object goes out of scope and it is used to free all resources allocated by the object during run-time. Destructor cannot be overloaded.

2.(b) Write the output of the following C++ code. Also, write the name of feature of Object Oriented Programming used in the following program jointly illustrated by the functions (i) to (iv):

2

```
#include <iostream.h>
void Line( )                                    //Function (i)
{ for (int L=1; L<=80; L++) cout<<"-";
  cout<<endl;
}
void Line(int N)                                //Function (ii)
{for(int L=1; L<=N; L++) cout<<"*"
 cout<<endl;
}
void Line(char C, int N)                        //Function (iii)
{for(int L=1;L<=N;L++) cout<<C;
  cout<<endl;
}
void Line(int M, int N)                         //Function (iv)
{for(int L=1; L<=n;L++) cout<<M*L;
  cout<<endl;
}
void main()
{int A=9,B=4,C=3;
char K='#';
Line(K,B);
Line(A,C);}
```

Ans.

# # # #

9 18 27

Function (i) to function (iv) represent function overloading (polymorphism).

2.(c.)Define a class Applicant in C++ with following description: (4)

Private Members

- **A data member ANo (Admission Number) of type long**
- **A data member Name of type string**
- **A data member Agg(Aggregate Marks) of type float**
- **A data member Grade of type char**
- **A member function GradeMe( ) to find the Grade as per the Aggregate Marks obtained by a student. Equivalent Aggregate marks range and the respective Grades are shown as follows**

Aggregate Marks	Grade
> = 80	A
Less than 80 and > = 65	B
Less than 65 and > = 50	C
Less than 50 and > = 33	D
Less than 33	E

Public Members

- **A function Enter( ) to allow user to enter values for ANo, Name, Agg & call function GradeMe( ) to find the Grade**
- **A function Result ( ) to allow user to view the content of all the data members.**

Ans.

```
class Applicant
{       long  ANo;
        char Name[25];
        float Agg;
        char Grade;
        void GradeMe( )
        {       if (Agg > = 80)
                        Grade = 'A';
                else if (Agg >= 65 && Agg < 80 )
                        Grade = 'B';
                else if (Agg >= 50 && Agg < 65 )
                        Grade = 'C;
                else if(Agg>=33 && Agg <50)
                        Grade = 'D';
                else
                        Grade= 'E';
        }
public:
        void Enter ( )
        {       cout <<"\n  Enter Admission No.       ";   cin>>ANo;
                cout <<"\n  Enter Name of  the Applicant      ";  cin.getline(Name,25);
                cout <<"\n  Enter Aggregate Marks obtained by the Candidate :";  cin>>Agg;
                GradeMe( );
        }

        void Result( )
        {       cout <<"\n  Admission No.     "<<ANo;
                cout <<"\n  Name of  the Applicant    ";<<Name;
                cout<<"\n  Aggregate Marks obtained by the Candidate.      " << Agg;
                cout<<\n    Grade Obtained  is    " << Grade ;
        }
};
```

2.(d) Answer the questions (i) to (iv) based on the following

4

```
class Student
{       int Rollno;
        char SName[20];
        float Marks;
protected:
        void Result( );
public:
        Student( );
        void Enroll ( );
        void Display ( );
};
class Teacher
{       long TCode;
        char TName [ 20];
protected :
        float Salary;
```

```
public :
        Teacher( );
        void Enter ( );
        void Show ( );
};
class Course : public Student, private Teacher
{       long CCode[10];
        char CourseName[50];
        char StartDate [8], EndDate[8];
        public:
        Course( );
        void Commence( );
        void CDetail( );
 };
```

(i) Write the names of member functions, which are accessible from objects of class Course

(ii) Write the names of all data members, which is/are accessible from member function Commence of class Course

(iii) Write the names of all the members, which are accessible from objects of class teacher.

(iv) Which type of inheritance is illustrated in the above C++ code?

Solution

(i). void Commence( ), void CDetail( ), void Enroll ( ), void Display ( );
(ii) CCode, CourceName, StartDate, EndDate, Salary,
(iii) void Enter ( ), void Show ( );
(iv) Multiple inheritance

3. (a) Write a Get2From1() function in C++ to transfer the content from one array ALL[ ] to two different arrays Odd[ ] and Even[ ]. The Odd[ ] array should contain the values from odd positions(1,3,5,….) of ALL[ ] and Even[ ] array should contain the values from even positions (0,2,4,…) of ALL[ ].

Example: If the ALL array contains 12,34,56,67,89,90 the Odd[ ] array should contain 34, 67,90 and the Even[ ] array should contain 12,56,89 **3**

Ans.

```
void Get2From1(int ALL[ ],int n, int Odd[ ],int Even[ ])
{int j=0,k=0;
for (int i=0; i<n; i++)
        if(i%2==0)
                Even[j++]=a[i];
        else
                Odd[k++]=a[i];
}
```

3. (b) An array G[50][20] is stored in the memory along the row with each of its elements occupying 8 bytes. Find out the location of G[10][15], if G[0][0] is stored at 4200. **3**

Ans.

Address of G[i][j]=address of G[0][0] +(i*number of columns present in array +j)*sizeof(element)
Address of G[10][15]=4200+ (10*20+15)*8=4200+215*8=4200+1720=5920

3. (c) Write a function in C++ to perform Delete operation on a dynamically allocated Queue containing Members details as given in the following definition of NODE: **4**

```
struct NODE
{long Mno;              //Member Number
 char Mname[20];        //Member Name
```

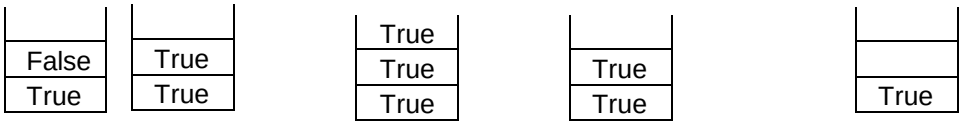
```
 NODE *Link;
};
class Queue
{NODE *front, *rear;
Queue() {front=NULL; rear=NULL;}
public:
void Addq();
void Delete();
};
void Queue::Delete()
{if(front==NULL)
        cout<<"Queue is empty";
else
        {NODE *t = front;
         front = front->Link;
         if(front==NULL)
                rear = NULL;
        delete t;
        }
}
```

3. (d) Write a DSUM() function in C++ to find sum of Diagonal Elements from a NxN Matrix. (Assuming that the N is a odd number)
2

```
Ans.
int DSUM(int A[ ],int N)
{int i, dsum1=0,dsum2=0;
for(i=0;i<N;i++)
        {dsum1+=A[i][i];
         dsum2+=A[N-(i+1)][i];
        }
return (dsum1+dsum2-A[N/2][N/2]);//because middle element is added twice
}
```

3. (e) Evaluate the following postfix notation of expression.
True, False, NOT, AND, True, True, AND, OR
2

NOT AND AND OR



ans: True

4. (a) Observe the program segment given below carefully and fill the blanks marked as statement1 and statement2 using seekg(), seekp(), tellp() and tellg() functions performing the required task.
1

```
#include<fstream.h>
class ITEM
{int Ino;char Iname[20];float price;
public:
void ModifyPrice();//the function is to modify price of a particular ITEM
};
void Item::ModifyPrice()
{fstream File;
File.open("ITEM.DAT",ios::binary|ios::in|ios::out);
```

```
int CIno;
cout<<"Item No to modify price:"; cin>>CIno;
while(File.read((char *)this, sizeof(ITEM)))
        {if(CIno==Ino)
                {cout<<"present Price :"<<Price<<endl;
                 cout<<"chaged Price :"; cin>> Price;
                int Filepos=_____;                      //Statement1
                _____;                                  //Statement2
                File.write((char *)this,sizeof(ITEM));
                }
        }
File.close();
}
```

Ans.

Statement1: int Filepos=File.tellg();

Statement2: File.seekp(Filepos-sizeof(ITEM), ios::beg);

4.(b) Write a function in C++ to count the no. of "He", or "She" words present in a text file "story.txt".

2

If the file "story.txt" content is as follows:

He is playing in the ground. She is
playing with her dolls.

The output of the function should be

Count ofHe/She in file is: 2

Ans.

```
#include<fstream.h>
#include<string.h>
#include<process.h>
int count()
{ifstream ifile("story.txt");
 if(!ifile)
        {       cout<<"could not open story.txt file "; exit(-1);       }
else
        {char s[20]; int c=0;
        while ( !ifile.eof() )
                {ifile>>s;
                 if ((strcmp(s,"He")==0)||(strcmp(s,"She")==0))
                        c++;
                }
        ifile.close();
        cout<<"Count of He/She in file is :"<<count;
        return c;
        }
}
```

4.(c) Write a function in C++ to search for camera from a binary file "CAMERA.DAT" containing the objects of class CAMERA (as defined below). The user should enter the ModelNo and the function should search and display the details of the camera.

3

```
class CAMERA
{long ModelNo; float MegaPixel; int Zoom; Char Details[120];
public:
void Enter()   {       cin>>ModelNo>>MegaPixel>>Zoom;   gets(Details);    }
void Display(){        cout<<ModelNo<<MegaPixel<<Zoom<<Details<<endl;  }
long GetModelNo() { return ModelNo;      }
```

Ans.

```
void search(long MNo)
{ifstream ifile ("CAMERA.DAT", ios::in | ios::binary);
 if ( ! ifile )
        {       cout<<"could not open CAMERA.DAT file ";  exit(-1); }
else
        {CAMERA c; int found=0;
        while(ifile.read((char *)&c, sizeof(c)))
                {if(c.GetModelNo()==MNo)
                        {c.Display(); found=1; break;}
                }
        }
if(found==0) cout<<"given ModelNo not found";
}
```

5. (a) What do you understand by Selection and Projections in relational algebra ?
2

Consider the following table EMPLOYEE and salgrade and answer (b) and (c) part of the Question:

Table: EMPLOYEE

ECODE	NAME	DESIG	SGRADE	DOJ	DOB
101	Abdul Ahmad	EXECUTIVE	S03	23-Mar-2003	13-Jan-1980
102	Ravi Chander	HEAD-IT	S02	12-Feb-2010	22-Jul-1987
103	John Ken	RECEPTIONIST	S03	24-Jan-2009	24-Feb-1983
104	Nazar Ameen	GM	S02	11-Aug-2006	03-Mar-1984
105	Priyam Sen	CEO	S01	29-Dec-2004	19-Jan-1982

Table:SALGRADE

SGRADE	SALARY	HRA
S01	56000	18000
S02	32000	12000
S03	24000	8000

(b) Write SQL commands for the following statements:
4

(i) To display the details of all EMPLOYEES in descending order of DOJ.
(ii) To display NAME and DESIG of those EMPLOYEES whose SALGRADE is either S02 or SO3.
(iii) To display the content of all the EMPLOYEES table, whose DOJ is in between 09-Feb-2006 and 08-Aug-2009.
(iv) To add a new row with the following
109, 'Harish Roy', 'HEAD-IT', 'S02','09-Sep-2007', '21-Apr-1983'

(c) Give the output of the following SQL queries :
2

(i) SELECT COUNT(SGRADE), SGRADE FROM EMPLOYEE GROUP BY SGRADE;
(ii) SELECT MIN(DOB), MAX(DOJ) FROM EMPLOYEE;

(iii) SELECT NAME,SALARY FROM EMPLOYEE E, SALGRADE S
WHERE E.SGARDE=S.SGRADE AND E.ECODE<103;
(iv) SELECT SGRADE, SALARY+HRA FROM SALGRADE WHERE SGRADE='S02'

Ans. Selection means selecting some rows(touples) from a relation according to given condition
e.g. $\sigma_{price>20.0}$(Item)
Project operation yields a vertical subset of a given relation(i.e. select all tuples containing only given columns of a relation).
e.g. $\pi_{NAME, DESIG}$(Employee)
(b)(i) SELECT * FROM EMPLOYEE ORDER BY DOJ DESC;
(ii) SELECT NAME,DESIG FROM EMPLOYEE WHERE SALGRADE IN('S02',S03');
(iii) SELECT * FROM EMPLOYEE WHERE DOJ BETWEEN '09-Feb-2006' AND '08-Aug-2009';
(iv) INSERT INTO EMPLOYEE
VALUES(109,'Harish Roy','HEAD-IT','S02','09-Sep-2007','21-Apr-1983');

(c) (i)

COUNT	SGRADE
2	S03
2	S02
1	S01

(ii) 13-Jan-1980 12-Feb-2010

(iii)

NAME	SALARY
Abdul Ahmad	24000
Ravi Chander	32000

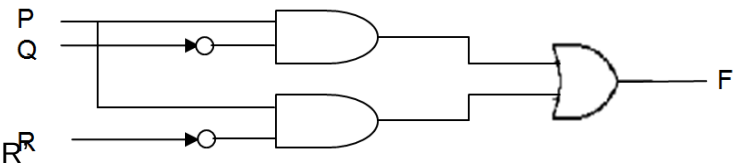
(iv)

SGRADE	SALARY+HRA
S02	44000

6.(a) Verify the following using truth table: **2**
X+Y.Z=(X+Y).(X+Z)
Ans.

X	Y	Z	Y.Z	**X+YZ**	(X+Y)	(X+Z)	**(X+Y)(X+Z)**
0	0	0	0	**0**	0	0	**0**
0	0	1	0	**0**	0	1	**0**
0	1	0	0	**0**	1	0	**0**
0	1	1	1	**1**	1	1	**1**
1	0	0	0	**1**	1	1	**1**
1	0	1	0	**1**	1	1	**1**
1	1	0	0	**1**	1	1	**1**
1	1	1	1	**1**	1	1	**1**

6.(b) Write the equivalent Boolean Expression for the following Logic Circuit : **2**

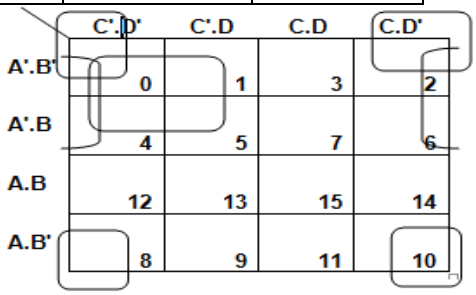


Ans. F=P.Q'+P.R

U	V	W	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0

6.(c) Write the SOP form of a Boolean function F, which is represented in a truth table as follows: **1**

1	0	1	0
1	1	0	1
1	1	1	1



Ans.
F=U'V'W'+U'VW+UVW'+UVW

6. (d) Reduce the Boolean expression using K-Map: 3

F(A,B,C,D)=∑(0,1,2,4,5,6,8,10)

Ans. F(A,B,C,D)=B1+B2+B3
B1=m0+m2+m8+m10=B'D'
B2=m0+m1+m4+m5=A'C'
B3=m0+m4+m2+m6=A'D'
F(A,B,C,D)=B'D'+A'C'+A'D'

7.(a) In networking, what is WAN? How is it different from LAN? **1**

Ans.

WAN is Wide Area Netork	LAN is Local Area Network
Span across countries Low data rate	Diameter of not more than a few kilometer High data rate

7.(b) Differentiate between XML and HTML. **1**

Ans. In HTML(HyperText Markup Language), both tag semantics and the tag set are fixed whereas, XML (eXtensible Markup Language) is a meta-language for describing markup languages, XML provides facility to define tags and the structural relationships between them. All the semantics of an XML document will either be defined by the applications that process them or by stylesheets.

7.(c) What is WEB2.0 ? **1**

Ans. The term **Web 2.0** is associated with web applications that facilitate participatory information sharing, interoperability, user-centered design, and collaboration on the World Wide Web. A Web 2.0 site allows users to interact and collaborate with each other in a social media dialogue as creators (prosumers) of user-generated content in a virtual community, in contrast to websites where users (consumers) are limited to the passive viewing of content that was created for them. Examples of Web 2.0 include social networking sites, blogs, wikis, video sharing sites,hosted services, web applications, mashups and folksonomies.

7.(d) Out of the following, identify client side script(s) and server side script(s) **1**

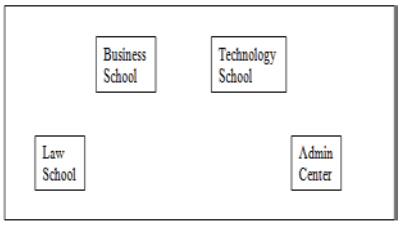
(i) Javascript (ii) ASP (iii) vbscript (iv) JSP

Ans. Client side scripts are Javascript, and vbscript, server side scripts are ASP, and JSP

7.(e) Great Sudies University is setting up its Academic school at Sunder Nagar and planning to set up a network. The university has 3 academic schools and one administration center as shown in the diagram bellow: **4**

Law school to Business school	60m

Law school to Technology School	90m
Law school to Admin Center	115m
Business school to Technology School	40m
Business school to Admin Center	45m
Technology school to Admin Center	25m

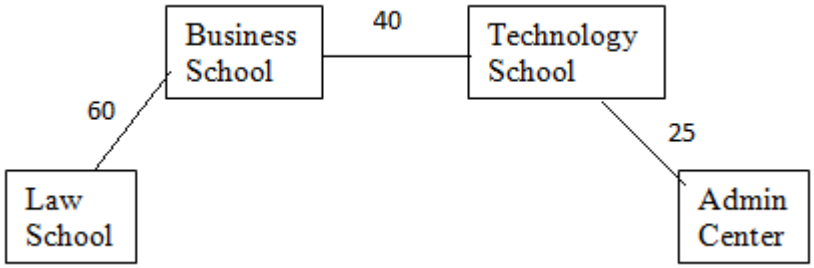


Law school	25
Technology school	50
Admin center	125
Business school	35

(i)Suggest the most suitable place(i.e school/center) to install the server of this university Sugewith a suitable reason.
(ii)Suggest an ideal layout for connecting these school/center for a wired connectivity.
(iii)Which device will you suggest to be placed/installed in each of these school/center to **efficiently** connect all the computers within these school/center.
(iv)The university is planning to connect its admission office in the closest big city,which is more than 350 km from the university.Which type of network out of LAN,MAN or WAN will be formed?Justify your answer.

(f)Compare Open Source Software and Proprietary Software.
(g)What are cookies?

Ans.
(i)Admin Center because it contains maximum number of computers (using 80-20 rule).
(ii) BUS topology is the best suitable cable layout.



(iii)Switch
(iv)WAN because LAN and MAN cant not spanmore than 100 km.

(f)Compare Open Source Software and Proprietar Software. **1**
Ans. Open Source Software can be freely used (source code is available to yhe customer) but it does not have to free of charge
Proprietary Software is the software that is neither open nor freely available(source code is not available,further distribution and modification is either forbidden or requires special permission by the supplier or vendor.
(g)What are cookies? **1**
Cookies are messages that a web server transmits to a web browser so that the web server can keep track of users activity on a specific web site.

Series SMA **Code No. 91**

Candidates must write the Code on the title page of the answer-book.

Roll No. | | | | | | | |

- Please check that this question paper contains **15** printed pages.
- Code number given on the right hand side of the question paper should be written on the title page of the answer-book by the candidate.
- Please check that this question paper contains **7** questions.
- **Please write down the Serial Number of the question before attempting it.**
- 15 minutes time has been allotted to read this question paper. The question paper will be distributed at 10.15 a.m. From 10.15 a.m. to 10.30 a.m., the students will read the question paper only and will not write any answer on the answer-book during this period.

COMPUTER SCIENCE

Time allowed : 3 hours *Maximum Marks : 70*

Instructions :

(i) ***All*** *questions are compulsory.*

(ii) *Programming Language : C++*

1. (a) Give the difference between the type casting and automatic type conversion. Also, give a suitable C++ code to illustrate both. *2*

(b) Which C++ header file(s) are essentially required to be included to run/execute the following C++ source code (Note : Do not include any header file, which is/are not required) : *1*

```
void main()
{
 char TEXT[]="SomeThing";
 cout<<"Remaining SMS Chars :"<<160-strlen(TEXT)<<endl;
}
```

(c) Rewrite the following program after removing the syntactical errors (if any). Underline each correction. 2

```
#include <iostream.h>
Class Item
{
  long IId,Qty;
public:
  void Purchase{cin>>IId>>Qty;}
  void Sale()
  {
    cout<<setw(5)<<IId<<" Old:"<<Qty<<endl;
    cout<<"New:"<<--Qty<<endl;
  }
};
void main()
{
  Item I;
  Purchase();
  I.Sale();
  I.Sale()
}
```

(d) Find the output of the following program : 3

```
#include <iostream.h>
class METRO
{
  int Mno,TripNo,PassengerCount;
public:
  METRO(int Tmno=1) {Mno=Tmno;TripNo=0;PassengerCount=0;}
  void Trip(int PC=20){TripNo++;PassengerCount+=PC;}
  void StatusShow(){cout<<Mno<<":"<<TripNo<<":"
                                <<PassengerCount<<endl;}
};
```

91 2

```
void main()
{
   METRO M(5),T;
   M.Trip();
   T.Trip(50);
   M.StatusShow();
   M.Trip(30);
   T.StatusShow();
   M.StatusShow();
}
```

(e) Find the output of the following program : 2

```
#include <iostream.h>
#include <ctype.h>
typedef char Str80[80];
void main()
{
   char *Notes;
   Str80 Str="vR2GooD";
   int L=6;
   Notes=Str;
   while (L>=3)
   {
      Str[L]=(isupper(Str[L])?tolower(Str[L]):
                                    toupper(Str[L]));
      cout<<Notes<<endl;
      L--;
      Notes++;
   }
}
```

(f) Observe the following program and find out, which output(s) out of (i) to (iv) will ***not*** be expected from the program ? What will be the minimum and the maximum value assigned to the variable Chance ? 2

```
#include <iostream.h>
#include <stdlib.h>
void main()
{
   randomize();
   int Arr[]={9,6},N;
   int Chance=random(2)+10;
   for (int C=0;C<2;C++)
   {
      N=random(2);
      cout<<Arr[N]+Chance<<"#";
   }
}
```

(i) 9#6#

(ii) 19#17#

(iii) 19#16#

(iv) 20#16#

2. (a) What is the difference between the members in private visibility mode and the members in protected visibility mode inside a class ? Also, give a suitable C++ code to illustrate both. 2

(b) Answer the questions (i) and (ii) after going through the following class : 2

```
class Travel
{
   int PlaceCode;char Place[20];float Charges;
public:
   Travel()                              //Function 1
   {
      PlaceCode=1;strcpy(Place,"DELHI");Charges=1000;
   }
   void TravelPlan(float C)              //Function 2
   {
      cout<<PlaceCode<<":"<<Place<<":"<<Charges<<endl;
   }
   ~Travel()                             //Function 3
   {
      cout<<"Travel Plan Cancelled"<<endl;
   }
   Travel(int PC,char P[],float C)       //Function 4
   {
      PlaceCode=PC;strcpy(Place,P);Charges=C;
   }
};
```

(i) In Object Oriented Programming, what are **Function 1** and **Function 4** combined together referred as ?

(ii) In Object Oriented Programming, which concept is illustrated by **Function 3** ? When is this function called/invoked ?

(c) Define a class RESTRA in C++ with following description : 4

Private Members

- FoodCode of type int
- Food of type string
- FType of type string
- Sticker of type string
- A member function GetSticker() to assign the following values for Sticker as per the given FType :

FType	Sticker
Vegetarian	GREEN
Contains Egg	YELLOW
Non-Vegetarian	RED

Public Members

- A function GetFood() to allow user to enter values for FoodCode, Food, FType and call function GetSticker() to assign Sticker.
- A function ShowFood() to allow user to view the content of all the data members.

(d) Answer the questions (i) to (iv) based on the following : 4

```
class COMPANY
{
   char Location[20];
   double Budget,Income;
protected:
   void Accounts();
public:
   COMPANY();
   void Register();
   void Show();
};
class FACTORY:public COMPANY
{
   char Location[20];
   int Workers;
protected:
   double Salary;
   void Computer();
public:
   FACTORY();
   void Enter();
   void Show();
};
class SHOP: private COMPANY
{
   char Location[20];
   float Area;
   double Sale;
public:
   SHOP();
   void Input();
   void Output();
};
```

(i) Name the type of inheritance illustrated in the above C++ code.

(ii) Write the name of data members, which are accessible from member functions of class SHOP.

(iii) Write the names of all the member functions, which are accessible from objects belonging to class FACTORY.

(iv) Write the names of all the members, which are accessible from objects of class SHOP.

3. (a) Write a function SWAP2BEST (int ARR[], int Size) in C++ to modify the content of the array in such a way that the elements, which are multiples of 10 swap with the value present in the very next position in the array. *3*

For example :

If the content of array ARR is

90, 56, 45, 20, 34, 54

The content of array ARR should become

56, 90, 45, 34, 20, 54

(b) An array T[20][10] is stored in the memory along the column with each of the elements occupying 2 bytes. Find out the memory location of T[10][5], if the element T[2][9] is stored at the location 7600. *3*

(c) Write a function in C++ to perform Insert operation in a static circular Queue containing Book's information (represented with the help of an array of structure BOOK). *4*

```
struct BOOK
{
    long Accno;            //Book Accession Number
    char Title[20];        //Book Title
};
```

(d) Write a function `ALTERNATE (int A[][3], int N, int M)` in C++ to display all alternate elements from two-dimensional array A (starting from `A[0][0]`). *2*

For example :

If the array is containing :

```
23   54   76
37   19   28
62   13   19
```

The output will be

```
23 76 19 62 19
```

(e) Evaluate the following POSTFIX notation. Show status of Stack after every step of evaluation (i.e. after each operator) : *2*

```
True,False,NOT,AND,False,True,OR,AND
```

4. (a) Observe the program segment given below carefully and the questions that follow : *1*

```
class Stock
{
  int Ino,Qty; char Item[20];
public:
  void Enter(){cin>>Ino;gets(Item); cin>>Qty;}
  void Issue(int Q){Qty+=Q;}
  void Purchase(int Q){Qty-=Q;)
  int GetIno(){return Ino;}
};
void PurchaseItem(int Pino,int PQty)
{
  fstream File;
  File.open("STOCK.DAT",ios::binary|ios::in|ios::out);
  Stock S;
  int Success=0;
```

```
while (Success==0 && File.read((char*)&S,sizeof(S)))
{
     if (Pino==S.GetIno())
     {
       S.Purchase(PQty);
       ____________________          //Statement 1
       ____________________          //Statement 2
       Success++;
     }
}
if (Success==1)
    cout<<"Purchase Updated"<<endl;
    else
      cout<<"Wrong Item No"<<endl;
    File.close();
}
```

(i) Write statement 1 to position the file pointer to the appropriate place, so that the data updation is done for the required item.

(ii) Write statement 2 to perform the write operation so that the updation is done in the binary file.

(b) Write a function in C++ to read the content of a text file "DELHI.TXT" and display all those lines on screen, which are either starting with 'D' or starting with 'M'. 2

(c) Write a function in C++ to search for the details (Phoneno and Calls) of those Phones, which have more than 800 calls from a binary file "phones.dat". Assuming that this binary file contains records/objects of class Phone, which is defined below. *3*

```
class Phone
{
   char Phoneno[10];int Calls;
public:
   void Get(){gets(Phoneno); cin>>Calls;}
   void Billing(){cout<<Phoneno<<"#"<<Calls<<endl;}
   int GetCalls(){return Calls;}
};
```

5. (a) Give a suitable example of a table with sample data and illustrate Primary and Alternate Keys in it. *2*

Consider the following tables CARDEN and CUSTOMER and answer (b) and (c) parts of this question :

Table : CARDEN

Ccode	CarName	Make	Color	Capacity	Charges
501	A-Star	Suzuki	RED	3	14
503	Indigo	Tata	SILVER	3	12
502	Innova	Toyota	WHITE	7	15
509	SX4	Suzuki	SILVER	4	14
510	C Class	Mercedes	RED	4	35

Table : CUSTOMER

CCode	Cname	Ccode
1001	Hemant Sahu	501
1002	Raj Lal	509
1003	Feroza Shah	503
1004	Ketan Dhal	502

(b) Write SQL commands for the following statements : 4

(i) To display the names of all the silver colored Cars.

(ii) To display name of car, make and capacity of cars in descending order of their sitting capacity.

(iii) To display the highest charges at which a vehicle can be hired from CARDEN.

(iv) To display the customer name and the corresponding name of the cars hired by them.

(c) Give the output of the following SQL queries : 2

(i) SELECT COUNT(DISTINCT Make) FROM CARDEN;

(ii) SELECT MAX(Charges),MIN(Charges) FROM CARDEN;

(iii) SELECT COUNT(*), Make FROM CARDEN;

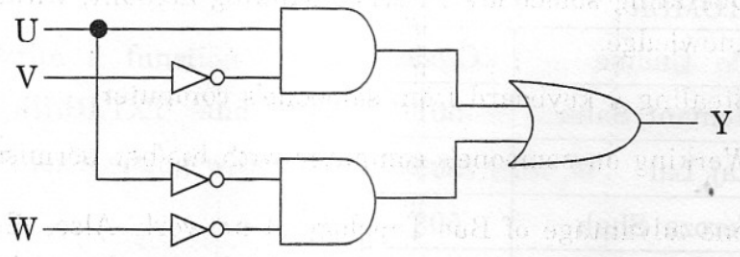
(iv) SELECT CarName FROM CARDEN WHERE Capacity=4;

6. (a) Verify the following using truth table : 2

(i) X.X′=0

(ii) X+1=1

(b) Write the equivalent Boolean expression for the following Logic Circuit : 2



(c) Write the SOP form of a Boolean function F, which is represented in a truth table as follows : *1*

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(d) Reduce the following Boolean Expression using K-Map : *3*

$F(A,B,C,D)=\Sigma(2,3,4,5,6,7,8,10,11)$

7. (a) What out of the following, will you use to have an audio-visual chat with an expert sitting in a far-away place to fix-up a technical issue ? *1*

(i) VoIP

(ii) email

(iii) FTP

(b) Name one server side scripting language and one client side scripting language. *1*

(c) Which out of the following comes under Cyber Crime ? *1*

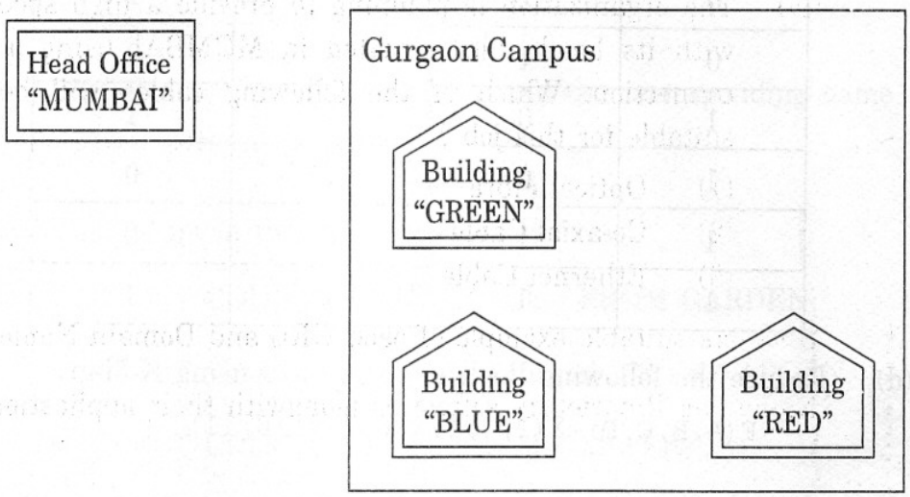
(i) Operating someone's Internet banking account, without his knowledge.

(ii) Stealing a keyboard from someone's computer.

(iii) Working on someone's computer with his/her permission.

(d) Write one advantage of Bus Topology of network. Also, illustrate how 4 computers can be connected with each other using star topology of network. *1*

(e) Workalot Consultants are setting up a secured network for their office campus at Gurgaon for their day-to-day office and web-based activities. They are planning to have connectivity between 3 buildings and the head office situated in Mumbai. Answer the questions (i) to (iv) after going through the building positions in the campus and other details, which are given below : 4



Distances between various buildings

Building "GREEN" to Building "RED"	110 m
Building "GREEN" to Building "BLUE"	45 m
Building "BLUE" to Building "RED"	65 m
Gurgaon Campus to Head Office	1760 KM

Number of Computers

Building "GREEN"	32
Building "RED"	150
Building "BLUE"	45
Head Office	10

(i) Suggest the most suitable place (i.e. building) to house the server of this organization. Also give a reason to justify your suggested location.

(ii) Suggest a cable layout of connections between the buildings inside the campus.

(iii) Suggest the placement of the following devices with justification :

(1) Switch

(2) Repeater

(iv) The organization is planning to provide a high speed link with its head office situated in MUMBAI using a wired connection. Which of the following cables will be most suitable for this job ?

(1) Optical Fibre

(2) Co-axial Cable

(3) Ethernet Cable

(f) Give one suitable example of each URL and Domain Name. *1*

(g) Name two Proprietary softwares alongwith their application. *1*

Series : SKS/1

Code No. 91/1

Roll No. | | | | | | | |

Candidates must write the Code on the title page of the answer-book.

- Please check that this question paper contains **15** printed pages.
- Code number given on the right hand side of the question paper should be written on the title page of the answer-book by the candidate.
- Please check that this question paper contains 7 questions.
- **Please write down the Serial Number of the question before attempting it.**
- 15 minutes time has been allotted to read this question paper. The question paper will be distributed at 10.15 a.m. From 10.15 a.m. to 10.30 a.m., the students will read the question paper only and will not write any answer on the answer-book during this period.

COMPUTER SCIENCE

Time allowed : 3 hours] *[Maximum Marks : 70*

Instructions : *(i) All questions are compulsory.*
(ii) Programming Language : C++

1. (a) What is the benefit of using function prototype for a function ? Give a suitable example to illustrate it using a C++ code. **2**

(b) Observe the following C++ code and write the name(s) of the header file(s), which will be essentially required to run it in a C++ compiler : **1**

```
void main ( )
{
    int Number;
    cin>>Number;
    if (abs (Number) ==Number);
    cout<<"Positive"<<endl;
}
```

(c) Observe the following C++ code carefully and rewrite the same after removing all the syntax error(s) present in the code. Ensure that you underline each correction in the code. 2

Important Note :

- All the desired header files are already included, which are required to run the code.
- Correction should not change the logic of the program.

```
#define Convert (P,Q) P+2*Q;
void main ()
{
    Float A, B, Result;
    cin>>A>>B ;
    Result=Convert [A,B] ;
    cout<<"Output:"<<Result<<endline;
}
```

(d) Observe the following C++ code carefully and obtain the output, which will appear on the screen after execution of it. 2

Important Note :

- All the desired header files are already included in the code, which are required to run the code.

```
void main ( )
{
    char *String="SHAKTI";
    int  *Point,Value[]={10,15,70,19};
    Point=Value;
    cout<<*Point<<String<<endl;
    String++;
    Point++;
    cout<<*Point<<String<<endl;
}
```

(f) Based on the following C++ code, find out the expected correct output(s) from the options (i) to (iv). Also, find out the minimum and the maximum value that can be assigned to the variable **Trick** used in the code at the time when value of **Count** is 3 : 2

```
void main()
{
    char Status[][10]={"EXCEL","GOOD","OK"};
    int Turn=10,Trick;
    for(int Count=1;Count<4;Count++)
    {
        Trick=random(Count);
        cout<<Turn-Trick<<Status[Trick]<<"#";
    }
}
```

(i) 10EXCEL#10EXCEL#8OK#

(ii) 10EXCEL#8OK#9GOOD#

(iii) 10EXCEL#9GOOD#10EXCEL#

(iv) 10EXCEL#10GOOD#8OK#

2. (a) Write any two differences between Constructor and Destructor. Write the function headers for constructor and destructor of a class Member. 2

(b) Answer the questions (i) and (ii) after going through the following class : 2

```
class Motor
{
    int MotorNo, Track;
    public:
        Motor();                //Function 1
        Motor(int MN);          //Function 2
        Motor(Motor &M);        //Function 3
        void Allocate();        //Function 4
        void Move();            //Function 5
};
```

```
void main()
{
    Motor M;
    :
    :
}
```

(i) Out of the following, which of the option is correct for calling Function 2 ?

```
Option 1 - Motor N(M);
Option 2 - Motor P(10);
```

(ii) Name the feature of Object Oriented Programming, which is illustrated by Function 1, Function 2 and Function 3 combined together.

(c) Define a class Tourist in C++ with the following specification : 4

Data Members

- Carno - to store Bus No
- Origin - to store Place name
- Destination - to store Place name
- Type - to store Car Type such as 'E' for Economy
- Distance - to store the Distance in Kilometers
- Charge - to store the Car Fare

Member Functions

- A constructor function to initialize Type as 'E' and Freight as 250
- A function CalcCharge() to calculate Fare as per the following criteria :

Type	Charge
'E'	16*Distance
'A'	22*Distance
'L'	30*Distance

- A function **Enter()** to allow user to enter values for Carno, Origin, Destination, Type and Distance. Also, this function should call **CalcCharge()** to calculate Fare.
- A function **Show()** to display the content of all the data members on screen.

```
void main()
{
    Motor M;
    :
    :
}
```

(i) Out of the following, which of the option is correct for calling Function 2 ?

```
Option 1 - Motor N(M);
Option 2 - Motor P(10);
```

(ii) Name the feature of Object Oriented Programming, which is illustrated by Function 1, Function 2 and Function 3 combined together.

(c) Define a class Tourist in C++ with the following specification : 4

Data Members

- Carno - to store Bus No
- Origin - to store Place name
- Destination - to store Place name
- Type - to store Car Type such as 'E' for Economy
- Distance - to store the Distance in Kilometers
- Charge - to store the Car Fare

Member Functions

- A constructor function to initialize Type as 'E' and Freight as 250
- A function CalcCharge() to calculate Fare as per the following criteria :

Type	Charge
'E'	16*Distance
'A'	22*Distance
'L'	30*Distance

- A function **Enter()** to allow user to enter values for Carno, Origin, Destination, Type and Distance. Also, this function should call **CalcCharge()** to calculate Fare.
- A function **Show()** to display the content of all the data members on screen.

```
void ChangeData()
{
    fstream File;
File.open("CUST.DAT",ios::binary|ios::in|ios::out);
    int Change=0,Location;
    long int ChangeCno;
    cout<<"Cno - whose email required to be modified:";
    cin>>ChangeCno;
    Customer CU;
    while(!Modify && File.read((char*)&CU,sizeof(CU)))
    {
        if (CU.GetCno()==ChangeCno)
        {
            CU.ModifyEmail();
            Location=File.tellg()- sizeof (CU);
    //Statement 1:To place file pointer to the
    required position
        _____;
    //Statement 2:To write the object CU on to the
    binary file
        _____;
        Change++;
        }
    }
    if (Change)
        cout<<"Email Modified... "<<endl;
    else
        cout<<"Customer not found... "<<endl;
    File.close();
}
```

(b) Write a function CountHisHer ( ) in C++ which reads the contents of a text file **diary.txt** and counts the words His and Her (not case sensitive). 2

For example, if the file contains :

```
Pinaky has gone to his friend's house. His friend's
name is Ravya. Her house is 12 KM from here.
```

The function should display the output as

```
Count for His:2
Count for Her:1
```

(c) Assuming the class VINTAGE as declared below, write a function in C++ to read the objects of VINTAGE from binary file **VINTAGE.DAT** and display those vintage vehicles, which are priced between 200000 and 250000. 3

```
class VINTAGE
{
    int VNO;          //Vehicle Number
    char VDesc[10];  //Vehicle Description
    float Price;
public :
    void GET() {cin>>VNO;gets (VDesc) ;cin>>Price;}
    void VIEW()
    {
    cout<<VNO<<endl;
    cout<<VDesc<<endl;
    cout<<Price<<endl;
    }
    float  ReturnPrice() {return  Price;}
};
```

5. (a) What is the difference between degree and cardinality of a table ? What is the degree and cardinality of the following table ? 2

Eno	Name	Salary
101	John Fedrick	45000
103	Raya Mazumdar	50600

NOTE :

Write SQL queries for (b) to (g) and write the outputs for the SQL queries mentioned shown in (h1) to (h4) parts on the basis of tables **ITEMS** and **TRADERS**

Table : ITEMS

CODE	INAME	QTY	PRICE	COMPANY	TCODE
1001	DIGITAL PAD 12i	120	11000	XENITA	T01
1006	LED SCREEN 40	70	38000	SANTORA	T02
1004	CAR GPS SYSTEM	50	21500	GEOKNOW	T01
1003	DIGITAL CAMERA 12X	160	8000	DIGICLICK	T02
1005	PEN DRIVE 32 GB	600	1200	STOREHOME	T03

Table: TRADERS

TCODE	TNAME	CITY
T01	ELECTRONIC SALES	MUMBAI
T03	BUSY STORE CORP	DELHI
T02	DISP HOUSE INC	CHENNAI

(b) To display the details of all the items in ascending order of item names (i.e. INAME). 1

(c) To display item name and price of all those items, whose price is in the range of 10000 and 22000 (both values inclusive). 1

(d) To display the number of items, which are traded by each trader. The expected output of this query should be : 1

```
T01 2

T02 2

T03 1
```

(e) To display the price, item name and quantity (i.e., qty) of those items which have quantity more than 150. 1

(f) To display the names of those traders, who are either from DELHI or from MUMBAI. 1

(g) To display the names of the companies and the names of the items in descending order of company names. 1

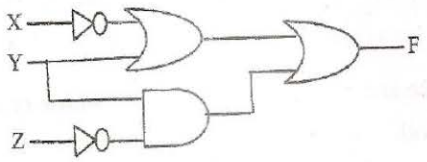
(h) Obtain the outputs of the following SQL queries based on the data given in tables ITEMS and TRADERS above. 2

```
h1) SELECT MAX(PRICE),MIN(PRICE) FROM ITEMS;

h2) SELECT PRICE*QTY AMOUNT
    FROM ITEMS WHERE CODE-1004;

h3) SELECT DISTINCT TCODE FROM ITEMS;

h4) SELECT INAME, TNAME
    FROM ITEMS I, TRADERS T
    WHERE I.TCODE=T.TCODE AND QTY<100;
```

6. (a) Verify the following using Boolean Laws : 2

```
A+C=A+A'.C+B.C
```

(b) Obtain the Boolean Expression for the logic circuit shown below : 2



(c) Write the Product of Sum form of the function G(U, V, W) for the following truth table representation of F : 1

U	V	W	G
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

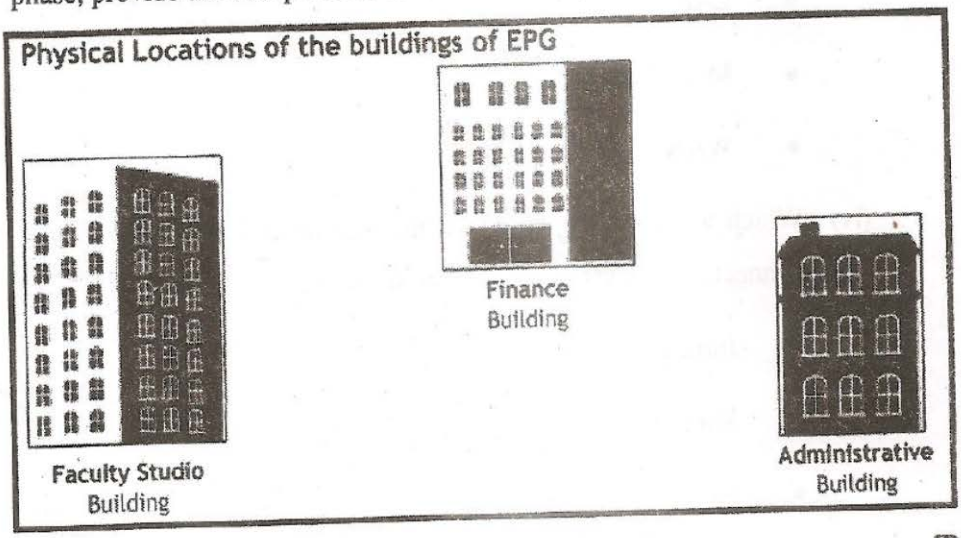
(d) Obtain the minimal form for the following Boolean expression using Karnaugh's Map. 3

$H(P,Q,R,S)=\Sigma\ (0,1,2,3,5,7,8,9,10,14,15)$

7. (a) What is the difference between domain name and IP address ? 1

(b) Write two advantages of using an optical fibre cable over an Ethernet cable to connect two service stations, which are 190m away from each other. 1

(c) Expertia Professional Global (EPG) is an online corporate training provider company for IT related courses. The company is setting up their new campus in Mumbai. You as a network expert have to study the physical locations of various buildings and the number of computers to be installed. In the planning phase, provide the best possible answers for the queries (i) to (iv) raised by them. 4



Building to Building distances (in Mtrs.)

From	To	Distance
Administrative Building	Finance Building	60
Administrative Building	Faculty Studio Building	120
Finance Building	Faculty Studio Building	70

Expected Computers to be installed in each Building :

Buildings	Computers
Administrative Building	20
Finance Building	40
Faculty Studio Building	120

(i) Suggest the most appropriate building, where EPG should plan to install the server.

(ii) Suggest the most appropriate building to building cable layout to connect all three buildings for efficient communication.

(iii) Which type of network out of the following is formed by connecting the computers of these three buildings ?

- LAN
- MAN
- WAN

(iv) Which wireless channel out of the following should be opted by EPG to connect to students of all over the world ?

- Infrared
- Microwave
- Satellite

(d) Write two advantages of using proprietary software over open source software. 1

(e) Which of the following crime(s) is/are covered under cybercrime ? 1

(i) Stealing brand new hard disk from a shop.

(ii) Getting into unknown person's social networking account and start messaging on his behalf.

(iii) Copying some important data from a computer without taking permission from the owner of the data.

Series : OSR/1

Roll No. | | | | | | | |

Candidates must write the Code on the title page of the answer-book.

- Please check that this question paper contains **11** printed pages.
- Code number given on the right hand side of the question paper should be written on the title page of the answer-book by the candidate.
- Please check that this question paper contains **7** questions.
- **Please write down the Serial Number of the question before attempting it.**
- 15 minutes time has been allotted to read this question paper. The question paper will be distributed at 10.15 a.m. From 10.15 a.m. to 10.30 a.m., the students will read the question paper only and will not write any answer on the answer-book during this period.

COMPUTER SCIENCE

Time allowed : 3 hours] *[Maximum Marks : 70*

Instructions : *(i) All questions are compulsory.*
(ii) Programming Language : C++

1. (a) What is the difference between actual and formal parameter ? Give a suitable example to illustrate using a C++ code. **2**

   (b) Observe the following C++ code and write the name(s) of the header file(s), which will be essentially required to run it in a C++ compiler : **1**

```
void main( )
{
    char Text[20],C;
    cin>>Text;
    C=tolower(Text[0]);
    Cout<<C<<" is the first char of "<<Text<<endl;
}
```

(c) Rewrite the following C++ code after removing all the syntax error(s), if present in the code. Make sure that you underline each correction done by you in the code. **2**

Important Note :

- Assume that all the required header files are already included, which are essential to run this code.
- The corrections made by you do not change the logic of the program.

```
typedef char[50] STRING;
void main( )
{
    City STRING;
    gets(City);
    cout<<City[0]<<' \ t<<City[2];
    cout<<City<<endline;
}
```

(d) Obtain the output from the following C++ program as expected to appear on the screen after its execution. **2**

Important Note :

- All the desired header files are already included in the code, which are required to run the code.

```
void main( )
{
    Char *String="SARGAM";
    int *Ptr, A[]={1,5,7,9};
    Ptr=A;
    cout<<*Ptr«String<<endl;
    String++;
    Ptr+=3;
    cout<<*Ptr<<String<<endl;
}
```

(e) Obtain the output of the following C++ program, which will appear on the screen after its execution.

Important Note : **3**

- All the desired header files are already included in the code, which are required to run the code.

```
Class Player
{
    int Score,Level;
    char Game;
```

```
public:
    Player(char GGame='A')
    {Score=0; Level=1; Game=GGame;}
    void Start(int SC);
    void Next();
    void Disp()
    {
        cout<<Game<<"@"<<Level<<endl;
        cout<<Score<<endl;
    }
};
void main()
{
    Player P,Q('B');
    P.Disp();
    Q.Start(75);
    Q.Next();
    P.Start(120);
    Q.Disp();
    P.Disp();
}
void Player::Next()
{
    Game=(Game=='A')?'B':'A';
}
void Player::Start(int SC)
{
    Score+=SC;
    If(Score>=100)
        Level=3;
    else if(Score>=50)
        Level=2;
    else
        Level=1;
}
```

(f) Read the following C++ code carefully and find out, which out of the given options (i) to (iv) are the expected correct output(s) of it. Also, write the maximum and minimum value that can be assigned to the variable Start used in the code : **2**

```
void main()
{
    int Guess[4]={200,150,20,250};
    int Start=random(2)+2;
    for (int C=Start;C<4;C++)
        cout<<Guess[C]<<"#";
}
```

(i) 200#150#
(ii) 150#20#
(iii) 150#20#250#
(iv) 20#250#

2. (a) Write 4 characteristics of a constructor function used in a class. **2**

(b) Answer the questions (i) and (ii) after going through the following class : **2**

```
class  Health
{
    int PId, DId;
public :
    Health(int PPId);          //Function 1
    Health();                  //Function 2
    Health (Health &H);        //Function 3
    void Entry();              //Function 4
    void Display();            //Function 5
};
void main ( )
{
    Health H(20);              //Statement 1
}
```

(i) Which of the function out of Function 1, 2, 3, 4 or 5 will get executed when the Statement 1 is executed in the above code ?

(ii) Write a statement to declare a new object G with reference to already existing object H using Function 3.

(c) Define a class CABS in C++ with the following specification: **4**

Data Members

- CNo - to store Cab No
- Type - to store a character 'A', 'B' or 'C' as City Type
- PKM - to store per Kilo Meter charges
- Dist - to store Distance travelled (in KM)

Member Functions

- A constructor function to initialize Type as 'A' and CNo as '1111'
- A function Charges( ) to assign PKM as per the following table:

Type	PKM
A	25
B	20
C	15

- A function **Register()** to allow administrator to enter the values for CNo and Type. Also, this function should call **Charges()** to assign PKM Charges.
- A function **ShowCab()** to allow user to enter the value of Distance and display CNo, Type, PKM, PKM*Distance (as Amount) on screen

(d) Consider the following C++ code and answer the questions from (i) to (iv) : **4**

```
Class Campus
{
    long Id;
    char City[20];
protected:
    char Country[20];
```

```
public :
    Campus();
    void Register();
    void Display();
};
class Dept: private Campus
{
    long DCode[10];
    char HOD[20];
protected :
    double Budget;
public:
    Dept();
    void Enter();
    void Show();
};
class Applicant: public Dept
{
    long RegNo;
    char Name[20];
public:
    Applicant();
    void Enroll();
    void View();
};
```

(i) Which type of Inheritance is shown in the above example ?

(ii) Write the names of those member functions, which are directly accessed from the objects of class Applicant.

(iii) Write the names of those data members, which can be directly accessible from the member functions of class Applicant.

(iv) Is it possible to directly call function Display( ) of class University from an object of class Dept ? (Answer as Yes or No).

3. (a) Write code for a function oddEven(`int s[],int N`) in C++, to add 5 in all the odd values and 10 in all the even values of the array S. **3**

Example: If the original content of the array S is

50	11	19	24	28

The modified content will be :

60	16	24	34	38

(b) An array T[25][20] is stored along the row in the memory with each element requiring 2 bytes of storage. If the base address of array T is 42000, find out the location of T[10][15]. Also, find the total number of elements present in this array. **3**

(c) Write a user-defined function **SumLast3(int A[][4],int N,int M)** in C++ to find and display the sum of all the values, which are ending with 3 (i.e., units place is 3). For example if the content of array is : **2**

33	13	92
99	3	12

The output should be

49

(d) Evaluate the following postfix expression. Show the status of stack after execution of each operation separately : **2**

F, T, NOT, AND, F, OR, T, AND

(e) Write a function POPBOOK( ) in C++ to perform delete operation from a Dynamic Stack, which contains Bno and Title. Consider the following definition of NODE, while writing your C++ code. **4**

```
struct NODE
{
    int Bno;
    char Title[20];
    NODE *Link;
};
```

4. (a) Fill in the blanks marked as Statement 1 and Statement 2, in the program segment given below with appropriate functions for the required task. **1**

```
class Medical
{
    int RNo;                    //Representative Code
    char Name[20];              //Representative Name
    char Mobile[12];            //Representative Mobile
public:
    void Input();//Function to enter all details
    void Show();//Function to display all details
    int RRno(){return RNo;}
    void ChangeMobile()//Function to change Mobile
    {
        cout<< "Changed Mobile:";
        gets(Mobile);
    }
```

```
};
void RepUpdate()
{
    fstream F;
    F.open("REP.DAT",ios::binary|ios::in|ios::out);
    int Change=0;
    int URno;
    cout<<"Rno(Rep No-to update Mobile):";
    cin>>URno;
    Medical M;
    while(!Change && F.read((char*)&M,sizeof(M)))
    {
        if(M.Rrno()==URno)
    {
    //Statement 1:To call the function to change Mobile No.
            _____;
    //Statement 2:To reposition file pointer to re-write
    //the updated object back in the file
            _____;
            F.write((char*)&M,sizeof(M));
            Change++;
        }
    }
    if (Change)
        cout<<"Mobile Changed for Rep "<<URno<<endl;
    else
        cout<<"Rep not in the Medical"<<endl;
    F.close();
}
```

(b) Write a function EUCount ( ) in C++, which should read each character of a text file IMP.TXT, should count and display the occurrence of alphabets E and U (including small cases e and u too). **2**

Example :

If the file content is as follows:

```
Updated information
is simplified by official websites.
```

The EUCount( ) function should display the output as

```
E:4
U:1
```

(c) Assuming the class GAMES as declared below, write a functions in C++ to read the objects of GAMES from binary file GAMES.DAT and display those details of those GAMES, which are meant for children of AgeRange "8 to 13". 3

```
Class GAMES
{
    int GameCode;
    char GameName[10];
    char AgeRange;
public :
    void Enter()
    {
        cin>>GameCode;
        gets(GameName);
        gets(AgeRange);
    }
    void Display()
    {
        cout<<GameCode<<":"<<GameName<<endl;
        cout<<AgeRange<<endl;
    }
    char* AgeR(){return AgeRange};
};
```

5. (a) Explain the concept of Union between two tables, with the help of appropriate example. 2

NOTE :

Answer the questions (b) and (c) on the basis of the following tables **STORE** and **ITEM**

Table: STORE

SNo	SName	Area
S01	ABC Computronics	GK II
S02	All Infotech Media	CP
S03	Tech Shoppe	Nehru Place
S04	Geeks Tecno Soft	Nehru Place
S05	Hitech Tech Store	CP

Table : ITEM

INo	IName	Price	SNo
T01	Mother Board	12000	S01
T02	Hard Disk	5000	S01
T03	Keyboard	500	S02
T04	Mouse	300	S01
T05	Mother Board	13000	S02
T06	Key Board	400	S03
T07	LCD	6000	S04
T08	LCD	5500	S05
T09	Mouse	350	S05
T10	Hard Disk	4500	S03

(b) Write the SQL queries (1 to 4) **4**

(1) To display IName and Price of all the Items in ascending order of their Price.

(2) To display SNo and SName of all Stores located in CP

(3) To display Minimum and Maximum Price of each IName from the table Item.

(4) To display IName, Price of all items and their respective SName where they are available.

(c) Write the output of the following SQL commands (1 to 4) : **2**

(1) SELECT DISTINCT INAME FROM ITEM WHERE PRICE >= 5000;

(2) SELECT AREA, COUNT(*) FROM STORE GROUP BY AREA;

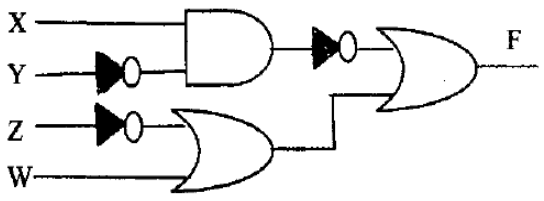
(3) SELECT COUNT(DISTINCT AREA) FROM STORE;

(4) SELECT INAME, PRICE * 0.05 DISCOUNT FROM ITEM WHERE SNO IN ('S02', 'S03');

6. (a) Name the law shown below and verify it using a truthtable. **2**

`A+B.C=(A+B).(A+C)`

(b) Obtain the Boolean Expression for the logic circuit shown below : **2**



(c) Write the Sum of Product form of the function F(P,Q,R) for the following truth table representation of F : 1

P	Q	R	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(d) Obtain the minimal form for the following Boolean expression using Karnaugh's Map. 3

$F(A,B,C,D) = \Sigma\ (1,4,5,9,11,12,13,15)$

7. (a) Write one characteristics each for 2G and 3G Mobile Technologies. 1

(b) What is the difference between Video Conferencing and Chat ? 1

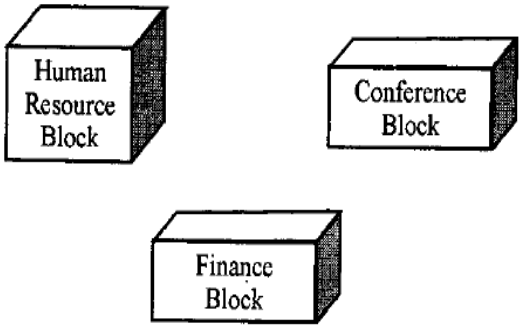
(c) Expand the following : 1

- GPRS
- CDMA

(d) Which type of network (out of LAN, PAN and MAN) is formed, when you connect two mobiles using Bluetooth to transfer a picture file. 1

(e) Trine Tech Corporation (TTC) is a professional consultancy company. The company is planning to set up their new offices in India with its hub at Hyderabad. As a network adviser, you have to understand their requirement and suggest them the best available solutions. Their queries are mentioned as (i) to (iv) below.

Physical Locations of the blocks of TTC



Block to Block distances (in Mtrs.)

Block (From)	Block (To)	Distance
Human Resource	Conference	110
Human Resource	Finance	40
Conference	Finance	80

Expected Number of Computers to be installed in each block

Block	Computers
Human Resource	25
Finance	120
Conference	90

(i) What will the most appropriate block, where TTC should plan to install their server ? 1

(ii) Draw a block to block cable layout to connect all the buildings in the most appropriate manner for efficient communication. 1

(iii) What will be the best possible connectivity out of the following, you will suggest to connect the new setup of offices in Bangalore with its London based office. 1

- Satellite Link
- Infrared
- Ethernet Cable

(iv) Which of the following device will be suggested by you to connect each computer in each of the buildings ? 1

- Switch
- Modem
- Gateway

(f) Write names of any two popular Open Source Software, which are used as operating system. 1

(g) Write any two important characteristics of Cloud Computing. 1
