केन्द्रीय विद्यालय-संगठन

KENDRIYA VIDYALAYA SANGATHAN

जम्मू-संभाग

JAMMU REGION

अध्ययन-सामग्री

STUDY MATERIAL 2012-13

कक्षा: 12

CLASS: 12

कंप्यूटर साइंस

COMPUTER SCIENCE

STUDY MATERIAL
FOR CLASS XII
COMPUTER SCIENCE
(2012-13)

PATRON

SH. A.S. GILL

Offg. Deputy Commissioner
KVS Jammu Region

GUIDANCE

SH. B.L. MORODIA

Asstt. Commissioner
KVS Jammu Region

COORDINATOR

DR. CHITRA MISHRA

Principal
KV No. 1 Jammu

RESOURCE PERSONS

SH. SUNIL KUMAR, PGT(CS) K.V No.1 Jammu
SMT. SONIKA CHIB , PGT(CS) K.V No.2 Jammu
SH. TARSAM KUMAR, PGT(CS) K.V Bantalab
SH. KIRPAL SINGH, PGT(CS) K.V No.1 Udhampur.

PREFACE

It gives me immense pleasure to present the Study Material of Class XII Computer Science for session 2012-13 by KVS Jammu Region.

This study material is written according to CBSE Syllabus of Computer Science for Class XII.

I am confident that the Study Material for Class XII Computer Science will help the students immensely to understand the concepts and will improve the quality performance of the students.

Wish you all the best .

(A. S. Gill)

Offg Deputy Commissioner

KVS RO, Jammu

CBSE Marks Distribution for Different Topics (Important Lessons)

SNo	Unit Name	Marks
1	**UNIT 1 Programming in C++**	30
2	**UNIT 2 Data Structures**	14
3	**UNIT 3 Database and SQL**	08
4	**UNIT 4 Boolean Logic**	08
5	**UNIT 5 Communication and Open source concepts**	10
	Total Marks	**70**

Weightage to different topics/content units

S No.	Topic	Marks
1	**Review of C++ covered in Class XI**	**12**
2	**Object Oriented Programming in C++**	**12**
3	**Data Structure & Pointers**	**14**
4	**Data File Handling in C++**	**06**
5	**Databases and SQL**	**08**
6	**Boolean Algebra**	**08**
7	**Communication and Open Source Concepts**	**10**
	Total	**70**

Weightage to different forms of questions

S. No.	Forms of Question	Marks for each question	No. of Questions	Total Marks
1	**Very Short Answer Questions (VSA)**	**01**	**09**	**09**
2	**Short Answer Questions- Type 1 (SA1)**	**02**	**13**	**26**
3	**Short Answer Questions- Type II (SAII)**	**03**	**05**	**15**
4	**Long Answer Questions- (LA)**	**04**	**05**	**20**
		Total	**32**	**70**

Difficulty Level of Questions

S. N.	Estimated Difficulty Level	Percentage of questions
1	**Easy**	**15%**
2	**Average**	**70%**
3	**Difficult**	**15%**

SUPPORT MATERIAL

INDEX

UNIT 1 : PROGRAMMING IN C++

Introduction to C++

- C++ programming language developed by AT&T Bell Laboratories in 1979 by Bjarne Stroustrup. C++ is fully based on O**bject Oriented Technology** i.e. C++ is ultimate paradigm for the modeling of information.
- C++ is the successor of C language.
- It is a case sensitive language.

Character Set- Set of characters which are recognized by c++compiler i.e Digits (0-9), Alphabets (A-Z & a-z) and special characters + - * , . " ' < > = { ( ]) space etc **i.e 256 ASCII characters.**

Tokens- Smallest individual unit. Following are the tokens

- **Keyword**-Reserve word having special meaning the language and can't be used as identifier.
- **Identifiers-**Names given to any variable, function, class, union etc. Naming convention (rule) for writing identifier is as under:
  i) First letter of identifier is always alphabet.
  ii) Reserve word cannot be taken as identifier name.
  iii) No special character in the name of identifier except under score sign '_'.
- **Literals**-Value of specific data type assign to a variable or constant. Four type of Literals:
  i) Integer Literal i.e **int x =10**
  ii) Floating point Literal i.e **float x=123.45**
  iii) Character Literal i.e **char x= 'a'**, **enclosed in single quotes and single character only.**
  iv) String Literal i.e cout<< **"Welcome"** , **anything enclosed in double quotes**
- **Operato**r – performs some action on data
  - Arithmetic(+,-,*,/,%)
  - Assignment operator (=)
  - Increment / Decrement (++, --)
  - Relational/comparison (<,>,<=,>=,==,!=).
  - Logical(AND(&&),OR(||),NOT(!).
  - Conditional (? :)

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

- **Punctuation –** used as separators in c++ **e.g. [{ ( ) }] , ; # = : etc**

Data type- A specifier to create memory block of some specific size and type. C++offers two types of data types:

1) **Fundamental type :** Which are not composed any other data type i.e. int, char, float and void
2) **Derived data type :** Which are made up of fundamental data type i.e array, function, class, union etc

Data type conversion- Conversion of one data type into another data type. Two type of conversion i.e

i) Implicit Conversion – It is automatically taken care by complier in the case of lower range to higher range e.g. **int x, char c='A' then x=c** is valid i.e character value in c is automatically converted to integer.
ii) Explicit Conversion- It is user-defined that forces an expression to be of specific type. **e.g. double x1,x2 and int res then res=int(x1+x2)**

Variable- Memory block of certain size where value can be stored and changed during program execution. **e.g. int x, float y, float amount, char c;**

Constant- Memory block where value can be stored once but can't changed later on during program execution.**e.g. const int pi =3.14;**

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(/*….*/)**
Cascading – Repeatedly use of input or output operators(">>" or "<<") in one statement with cin or cout.

Control structure:

Sequence control statement(if)	**conditional statement (if else)**	**Multiple Choice Statement If –else-if**	**Switch Statement (Alternate for if-else- if) works for only exact match**	**loop control statement (while ,do… while, for)**
Syntax	Syntax	Syntax	Syntax	Syntax

if(expression)
{
statements;
}
	If(expression)
	{
	
	statements;
	}
	else
	{
	
	statements;
	}
		If (expression)
		{
		statements
		}
		else
		if(expression)
		{
		statement
		}
		else
		{
		statement
		}
			switch(int / char variable)
			{
			case literal1:
			[statements
			break;]
			case literal2:
			[statements,
			break;]
			default:statements;
			}
			
			Break is compulsory statement with every **case** because if it is not included then the controls executes next case statement until next break encountered or end of **swtich** reached.
			Default is optional, it gets executed when no match is found
				while(expression)
				{
				statements;
				}
				Entry control loop works for true condition.
				
				do
				{
				statements;
				} **while(expression);**
				Exit Control Loop execute at least once if the condition is false at beginning.
				
				for loop
				for(expression1;expression2;expression3)
				{
				statement;}
				Entry control loop works for true condition and preferred for fixed no.of times.

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 terminate the program depending upon certain condition.

break- exit from the current loop depending upon certain condition.

continue- to skip the remaining statements of the current loop and passes control to the next loop control statement.

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	Nome Purpose
iostream.h	Defines stream classes for input/output streams
stdio.h	Standard input and output
ctype.h	Character tests
string.h	String operations
math.h	Mathematical functions such as sin() and cos()
stdlib.h	Utility functions such as malloc() and rand()

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 upercase 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.
- **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 element of same type that are referred by a common name.

One Dimensional array

- An array is a continuous memory location holding similar type of data in single row or single column. Declaration in c++ is as under:
  const int size =20;
  int a[size] or int a[20]. The elements of array accessed with the help of an index.
  For example : for(i=0;i<20;i++) cout<<a[i];
- **String (Array of characters)** –Defined in c++ as one dimensional array of characters as char s[80]= "Object oriented programming";

Two dimensional array

- A two diamensional array is a continuous memory location holding similar type of data arranged in row and column format (like a matrix structure).
  Declaration – int a[3][4], means 'a' is an array of integers are arranged in 3 rows & 4 columns.

Function -Name given to group of statements that does some specific task and may return a value. Function can be invoked(called) any no. of time and anywhere in the program.

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);

Actual Parameters
Variables associated with function name during function call statement.

Formal Parameters
Variables which contains copy of actual parameters inside the function definition.

Local variables

- Declared inside the function only and its scope and lifetime is function only and hence accessible only inside function.

Global variables

- Declared outside the function and its scope and lifetime is whole program and hence accessible to all function in the program from point declaration.

  Example :

```
#include <iostream.h>
int a=20;  // global
void main()
{
        int b=10;  // local
        cout<<a<<b;
}
```

Passing value to function-

- **Passing by value-** In this method separate memory created for formal arguments and if any changes done on formal variables , it will not affect the actual variables.So actual variables are preserved in this case
- **Passing by address/reference-** In this method no separate memory created for formal variables i.e formal variables share the same location of actual variables and hence any change on formal variables automatically reflected back to actual variables.

Example :

```
 void sample( int a, int &b)
{
  a=a+100;
b=b+200;
cout<<a<<b;
}
void main()
{
int a=50, b=40;
cout<<a<<b;   // output 50  40
sample(a,b)   // output 150  240
cout<<a<<b; // output   50  240
}
```

Function overloading

- Processing of two or more functions having same name but different list of parameters

Function recursion

- Function that call itself either directly or indirectly.

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

e.g. struct employee
{
int empno;
char name[30];
char design[20];
char department[20];
}
Declaration: employee e;
Input /Output : cin>>e.empno; // members are accessed using dot(.) operator.
cout<<e.empno;

Nested structure

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

e.g. struct address
{ int houseno;
char city[20];
char area[20];
long int pincode;}
struct employee
{
int empno;
char name[30];
char design[20];
char department[20];
address ad; // **nested structure**
}
Declaration: employee e;
Input /Output : cin>>e.ad.houseno; // members are accessed using dot(.) operator.
cout<<e.ad.houseno;

typedef

Used to define new data type name.
e.g. typedef char Str80[80]; Str80 str;

#define Directives

- Use to define a constant number or macro or to replace an instruction.

1 Marks questions

Which C++ header file(s) will be essentially required to be included to run /execute the following C++ code:

```
void main()
{
char Msg[ ]="Sunset Gardens";
for (int I=5;I<strlen(Msg);I++)
puts(Msg);
}
```

Ans : stdio.h, string.h

Name the header files that shall be need for the following code: (CBSE 2012)

```
void main()
{
  char text[] =”Something”
  cout<<”Remaining SMS chars: ”<<160-strlen(text)<<endl;
 }
```

Ans: iostream.h/iomanip.h , string.h

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()
}
```

Ans :

```
#include<iostream.h>
class Item                    // C capital
{
long IId, Qty;
public:
 void Purchase ( ) { cin>>IId>>Qty;}  // ( ) after function name
  void Sale( )
   {
     cout<<setw(5)<<IId<<"Old:"<< Qty<<endl;
     cout<< "New :"<<Qty<<endl;
   }};
void main()
{
Item I;
I. Purchase();  // object missing
I.Sale()  ;       // ; is missing
}
```

Either the statement is removed or header file included as #include<iomanip.h>

2) **Find the output of the following program:** **CBSE 2012**

```
#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++;
 }}
```

Ans : vR2Good
R2GoOd
2GOOd
gOOd

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?

CBSE 2012

```
#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#

Ans: The output not expected from program are (i),(ii) and (iv)
Minimum value of Chance =10
Maximum value of Chance = 11

3 Marks questions:

4) **Find the output of the following program:** **CBSE 2012**

```
#include<iostream.h>
class METRO
{
int Mno, TripNo, PassengerCount;
public:
METRO(int Tmno=1) { Mno =Tmno; PassengerCount=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.StatusShow();
M.StatusShow();
}
```

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

2& 3 marks practice questions:

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 Encript ( 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; } }
```

9) **Rewrite the following C++ program after removing the syntax error(s) if any. Underline each correction. [CBSE 2010]**

```
include<iostream.h>
class FLIGHT
{       Long FlightCode;
        Char Description[25];
public
        void addInfo()
        {       cin>>FlightCode; gets(Description);}
        void showInfo()
        {               cout<<FlightCode<<":"<<Description<<endl;}
};
void main( )
{       FLIGHT F;
        addInfo.F();
        showInfo.F;
}
```

10) In the following program, find the correct possible output(s)from the options:

```
#include<stdlib.h>
#include<iostream.h>
void main( )
{ randomize( );
char City[ ][10]={"DEL", "CHN", "KOL", "BOM", "BNG"};
int Fly;
for(int I=0; I<3;I++)
{Fly=random(2) + 1;
cout<<City[Fly]<< ":";
}}
```

Outputs:

(i) DEL : CHN : KOL: (ii) CHN: KOL : CHN:

(iii) KOL : BOM : BNG: (iv) KOL : CHN : KOL:

11) In the following C++ program what is the expected value of Myscore from options (i) to (iv) given below. Justify your answer.

```
#include<stdlib.h>
#include<iostream.h>
void main( )
{       randomize( );
int Score[ ] = {25,20,34,56,72,63},Myscore;
cout<<Myscore<<endl;
}
```

i) 25 (ii) 34 (iii) 20 (iv) Garbage Value.

Function overloading in C++

- 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;

  // Call print to print integer
  pd.print(5);
  // Call print to print float
  pd.print(500.263);
  // Call print to print character
  pd.print("Hello C++");

  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++
```

OBJECT ORIENTED PROGRAMMING CONCEPTS

Object Oriented Programming follows bottom up approach in program design and emphasizes on safety and security of data..

FEATURES OF OBJECT ORIENTED PROGRAMMING:

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.
- Derived class is also known as a child class or sub class. Inheritance helps in reusability of code , thus reducing the overall size of the program

Data Abstraction:

- It refers to the act of representing essential features without including the background details .Example : For driving , only accelerator, clutch and brake controls need to be learnt rather than working of engine and other details.

Data Encapsulation:

- It means wrapping up data and associated functions into one single unit called class..
- A class groups its members into three sections :public, private and protected, where private and protected members remain hidden from outside world and thereby helps in implementing data hiding.

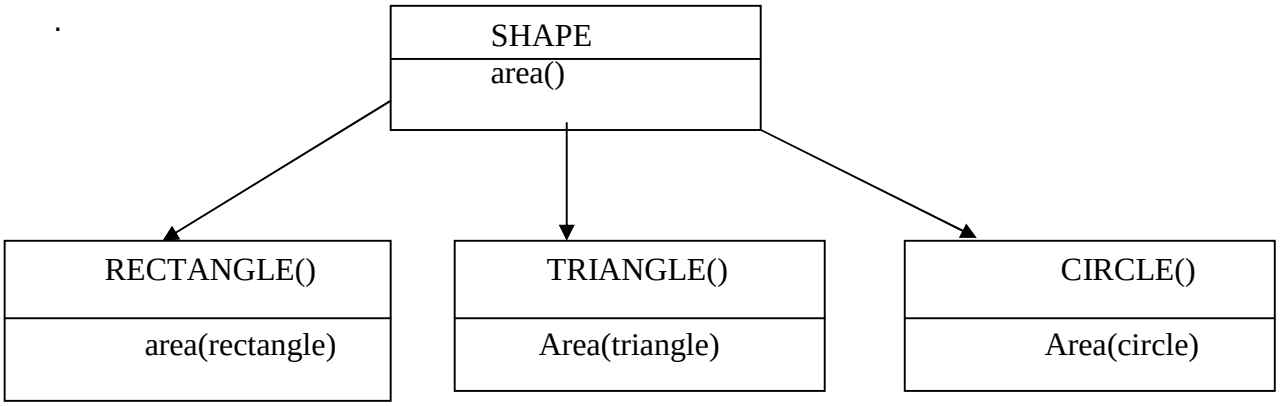
Modularity :

- The act of partitioning a complex program into simpler fragments called modules is called as modularity.
- It reduces the complexity to some degree and
- It creates a number of well defined boundaries within the program .

Polymorphism:

- **Poly** means many and **morphs** mean form, so polymorphism means one name multiple forms.
- It is the ability for a message or data to be processed in more than one form.
- C++ implements Polymorhism through Function Overloading , Operator overloading and Virtual functions .

.



Objects and Classes :

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

A **Class i**s a group of similar objects . **Objects** share two characteristics: They all have *state* and *behavior*. For example : Dogs have state (name, color, breed, hungry) and behavior (barking, fetching, wagging tail). Bicycles also have state (current gear, current pedal cadence, current speed) and behavior (changing gear, applying brakes). Identifying the state and behavior for real-world objects is a great way to begin thinking in terms of object-oriented programming. These real-world observations all translate into the world of object-oriented programming.

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state 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 just binds data and its associated functions under one unit there by enforcing encapsulation.**
- Classes define types of data structures and the functions that operate on those data structures.
- A 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 in Classes:

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

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 (.).

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

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

```
class sum
{
int A, B, Total;
public:
void getdata ();
void display ();
};
void sum:: getdata ()        // Function definition outside class definition Use of :: operator
{
cout<<" \n enter the value of A and B";
cin>>A>>B;
}
void sum:: display ()        // Function definition outside class definition Use of :: operator
{
Total =A+B;
cout<<"\n the sum of A and B="<<Total;
}
```

- **Inside 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 ()
{
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 and base classes are public by default. A class defined with the `class` keyword has private members and base classes by default. 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.**

Example

```
inline  void Square (int a)

{ cout<<a*a;}

void main()

{.

 Square(4);   ⟶   { cout <<4*4;}

 Square(8)  ;   ⟶   { cout <<8*8; }

}
```

In place of function call , function body is substituted because Square () is inline function

Pass Object As An Argument

/*C++ PROGRAM TO PASS OBJECT AS AN ARGUMEMT. The program Adds the two heights given in feet and inches. */

```
#include< iostream.h>
#include< conio.h>
class height
{
int feet,inches;
public:
void getht(int f,int i)
{
feet=f;
inches=i;
}
void putheight()
{
cout< < "\nHeight is:"< < feet< < "feet\t"< < inches< < "inches"< < endl;
}
void sum(height a,height b)
{
height n;
n.feet = a.feet + b.feet;
n.inches = a.inches + b.inches;
if(n.inches ==12)
{
n.feet++;
n.inches = n.inches -12;
}
cout< < endl< < "Height is "< < n.feet< < " feet and "< < n.inches< < endl;
}};
void main()
{height h,d,a;
clrscr();
h.getht(6,5);
a.getht(2,7);
h.putheight();
a.putheight();
d.sum(h,a);
getch();
}
```

```
/**********OUTPUT***********
Height is:6feet 5inches
Height is:2feet 7inches
Height is 9 feet and 0
```

4 Marks Solved Problems :

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%**
Up to 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 tax) //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; }
};
```

Q 2 : **Define a class HOTEL in C++ with the following description:**

Private Members

- Rno //Data Member to store Room No
- Name //Data Member to store customer Name
- Tariff //Data Member to store per day charge
- NOD //Data Member to store Number of days
- CALC //A function to calculate and return amount as NOD*Tariff and if the value of NOD*Tariff is more than 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 Rno, Name, Tariff, NOD and Amount (Amount to be displayed by calling function CALC( )

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	**D**

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
                Grade = 'D';
    }
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 ;
    }
};
```

Q 4 Define a class ITEM in C++ with following description:
Private members:

- **Icode of type integer (Item Code)**
- **Item of type string (Item Name)**
- **Price of type Float (Price of each item)**
- **Qty of type integer (Quantity in stock)**
- **Discount of type float (Discount percentage on the item)**
- **A find function finddisc( ) to calculate discount as per the following rule:**
  If Qty <=50 discount is 0%
  If 50 < Qty <=100 discount is 5%
  If Qty>100 discount is 10%

Public members :
A function Buy( ) to allow user to enter values for Icode, Item,Price, Qty and call function Finddisc ( ) to calculate the discount.
A function showall ( ) to allow user to view the content of all the data members.

Ans : class ITEM
{int Icode,Qty;
char item[20];
float price,discount;
void finddisc();
public:
void buy();
void showall();
};
void stock::finddisc()
{If (qty<=50)
Discount=0;
Else if (qty> 50 && qty <=100)
Discount=0.05*price;
Else if (qty>100)
Discount=0.10*price;
}
void stock::buy()
{cout<<"Item Code :";cin>>Icode;
cout<<"Name :";gets(Item);
cout<<"Price :";cin>>Price;
cout<<"Quantity :";cin>>Qty;
finddisc();
}
void TEST::DISPTEST()
{cout<<"Item Code :";cout<<Icode;
cout<<"Name :";cout<<Item;
cout<<"Price :";cout<<Price;
cout<<"Quantity :";cout<<Qty;
cout<<"Discount :";cout<<discount;
}

4 marks Practice Problems :

Q 1 Define a class **employee** with the following specifications : **4**

Private members of class employee

- empno integer
- ename 20 characters
- basic, hra, da float
- netpay float
- calculate() A function to calculate basic + hra + da with float return type

Public member function of class employee

- havedata() function to accept values for empno, sname, basic, hra, da and invoke calculate() to calculate netpay.
- dispdata() function to display all the data members on the screen.

Q2 Define a class **Student** with the following specifications : **4**

Private members :

- roll_no integer
- name 20 characters
- class 8 characters
- marks[5] integer
- percentage float

- Calculate() a function that calculates overall percentage of marks and return the percentage of marks.

public members :

- Readmarks() a function that reads marks and invoke the Calculate function.
- Displaymarks() a function that prints the marks.

Q3 : Define a class **repor**t with the following specification : **4**

Private members :

- adno 4 digit admission number
- name 20 characters
- marks an array of 5 floating point values
- average average marks obtained
- getavg() to compute the average obtained in five subjects

Public members :

- readinfo() function to accept values for adno, name, marks, and invoke the function getavg().
- displayinfo() function to display all data members on the screen you should give function definitions.

Q4 Declare a class **myfolder** with the following specification : **4**

Private members of the class

- Filenames – an array of strings of size[10][25](to represent all the names of files inside myfolder)
- Availspace – long (to represent total number of bytes available in myfolder)
- Usedspace – long (to represent total number of bytes used in myfolder)

public members of the class

- Newfileentry() – A function to accept values of Filenames, Availspace and Usedspace fromuser
- Retavailspace() – A Fucntion that returns the value of total Kilobytes available (1 Kilobytes = 1024 bytes)
- Showfiles() – a function that displays the names of all the files in myfolder

2 Marks Practice Problems

1. What is relation between class and object?
2. What are inline functions? Give example
3. Difference between private & public access specifiers.
4. How class implements data-hiding & encapsulation?
5. What is the difference between structure and a class ?
6. How is inline function different from a normal function ?

CONSTRUCTORS AND DESTRUCTORS

CONSTRUCTORS :

A member function with the same as its class is called Constructor and it is used to initialize the object of that class with a legal initial value.

Example :

```
class Student
{
  int rollno;
float marks;
public:
student( )              //Constructor
{
rollno=0;
marks=0.0;
}
                  //other public members
};
```

TYPES OF CONSRUCTORS:

1. Default Constructor:

A constructor that accepts no parameter is called the Default Constructor. 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.

2. Parameterized Constructors:

A constructor that accepts parameters for its invocation is known as parameterized Constructors , also called as Regular Constructors.

DESTRUCTORS:

- A destructor is also a member function whose name is the same as the class name but is preceded by tilde("~").It is automatically by the compiler when an object is destroyed. 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.

Example :

```
class TEST
{   int Regno,Max,Min,Score;
Public:
 TEST( )                          // Default Constructor
 {                        }
 TEST (int Pregno,int Pscore)     //  Parameterized Constructor
{
Regno = Pregno ;Max=100;Max=100;Min=40;Score=Pscore;
}
~ TEST ( )                        // Destructor
{ Cout<<"TEST Over"<<endl;}
};
```

The following points apply to constructors and destructors:

- Constructors and destructors do not have return type, not even void nor can they 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.
- Derived classes do not inherit constructors or destructors from their base classes, but they do call the constructor and destructor of base classes.
- The default destructor calls the destructors of the base class and members of the derived class.
- The destructors of base classes and members are called in the reverse order of the completion of their constructor:
- The destructor for a class object is called before destructors for members and bases are 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**.
- A copy constructor is a constructor of the form **classname(classname &).**The compiler will use the copy constructors whenever you initialize an instance using values of another instance of the same type.
- Copying of objects is achieved by the use of a copy constructor and a assignment operator.

Example :

```
class Sample{    int i, j;}
public:
Sample(int a, int b)        // constructor
{ i=a;j=b;}
Sample (Sample & s)      //copy constructor
{ j=s.j  ; i=s.j;
    Cout  <<"\n Copy constructor working \n";
}
void print (void)
{cout <<i<< j<< "\n";}
:
};
```

Note : ***The argument to a copy constructor is passed by reference, the reason being that when an argument is passed by value, a copy of it is constructed. But the copy constructor is creating a copy of the object for itself, thus ,it calls itself. Again the called copy constructor requires another copy so again it is called.in fact it calls itself again and again until the compiler runs out of the memory .so, in the copy constructor, the argument must be passed by reference.***

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

- **When an object is passed by value to a function**:
  The pass by value method requires a copy of the passed argument to be created for the function to operate upon .Thus to create the copy of the passed object, copy constructor is invoked
  If a function with the following prototype :
  void cpyfunc(Sample); // Sample is a class
  then for the following function call
  cpyfunc(obj1); // obj1 is an object of Sample type
  the copy constructor would be invoked to create a copy of the obj1 object for use by cpyfunc().

- **When a function returns an object** :
  When an object is returned by a function the copy constructor is invoked

  Sample cpyfunc(); // Sample is a class and it is return type of cpyfunc()

  If func cpyfunc() is called by the following statement

  obj2 = cpyfunc();

  Then the copy constructor would be invoked to create a copy of the value returned by cpyfunc() and its value would be assigned to obj2. The copy constructor creates a temporary object to hold the return value of a function returning an object.

1 & 2 Marks Solved Problems :

Q1 :- 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);

Q2 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, 'a9');
welcome obj1(50, '9');
welcome obj3();
obj1= welcome (45, 'T');
obj3= welcome;
```

Ans. Valid and invalid statements are

welcome obj (33, ‘a9’);	**valid**
welcome obj1(50, ‘9’);	**valid**
welcome obj3();	**invalid**
obj1= welcome (45, ‘T’);	**valid**
obj3= welcome;	**invalid**

2 Marks Practice Problems

Q1 What do you understand by constructor and destructor functions used in classes ? How are these functions different from other member functions ? 2

Q2 What do you understand by default constructor and copy constructor functions used in classes ? How are these functions different from normal constructors ? **2**

Q3 Given the following C++ code, answer the questions (i) & (ii). 2

```
class TestMeOut
{
public :
~TestMeOut() // Function 1
{ cout << "Leaving the examination hall " << endl; }
TestMeOut() // Function 2
{ cout << "Appearing for examination " << endl; }
void MyWork() // Function 3
{ cout << "Attempting Questions " << endl; }
};
```

(*i*) In Object Oriented Programming, what is Function 1 referred as and when does it get invoked / called ?

(*ii*) In Object Oriented Programming, what is Function 2 referred as and when does it get invoked / called ?

INHERITANCE

- **Inheritance is the process by which new classes called *derived* classes are created from existing classes called *base* classes**.
- The derived classes have all the features of the base class and the programmer can choose to add new features specific to the newly created derived class.
- The idea of inheritance implements the **is a** relationship. For example, mammal IS-A animal, dog IS-A mammal hence dog IS-A animal as well and so on.

Features or Advantages of Inheritance:

- ***Reusability of Code***
- ***Saves Time and Effort***
- ***Faster development, easier maintenance and easy to extend***
- ***Capable of expressing the inheritance relationship and its transitive nature which ensures closeness with real world problems .***

Base & Derived Classes:

A class can be derived from more than one classes, which means it can inherit data and functions from multiple base classes. A class derivation list names one or more base classes and has the form:

class derived-class: access-specifier base-class

Where access is one of **public, protected,** or **private**.

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*.

EXAMPLE OF SINGLE INHERITANCE

Consider a base class **Shape** and its derived class **Rectangle** as follows:

```
// Base class
class Shape
{
  public:
    void setWidth(int w)
    {
      width = w;
    }
    void setHeight(int h)
    {
      height = h;
    }
  protected:
    int width;
    int height;
};
```

```
// Derived class
class Rectangle: public Shape
{
  public:
    int getArea()
    {
      return (width * height);
    }
};
int main(void)
{
  Rectangle Rect;
   Rect.setWidth(5);
  Rect.setHeight(7);
  // Print the area of the object.
  cout << "Total area: " << Rect.getArea() << endl;

  return 0;

}
```

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

Total area: 35

Access Control and Inheritance:

A derived class can access all the non-private members of its base class. Thus base-class members that should not be accessible to the member functions of derived classes should be declared private in the base class.We can summarize the different access types according to who can access them in the following way:

Access	public	protected	private
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

A derived class inherits all base class methods with the following exceptions:

- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.
- The friend functions of the base class.

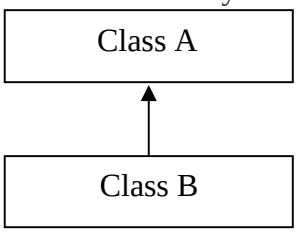
When deriving a class from a base class, the base class may be inherited through **public, protected** or **private** inheritance. We hardly use **protected** or **private** inheritance but **public** inheritance is commonly used. While using different type of inheritance, following rules are applied:

1. **Public Inheritance:** When deriving a class from a **public** base class, **public** members of the base class become **public** members of the derived class and **protected** members of the base class become **protected** members of the derived class. A base class's **private** members are never accessible directly from a derived class, but can be accessed through calls to the **public** and **protected** members of the base class.
2. **Protected Inheritance:** When deriving from a **protected** base class, **public** and **protected** members of the base class become **protected** members of the derived class.
3. **Private Inheritance:** When deriving from a **private** base class, **public** and **protected** members of the base class become **private** members of the derived Class.

TYPES OF INHERITANCE

1. Single class Inheritance:

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

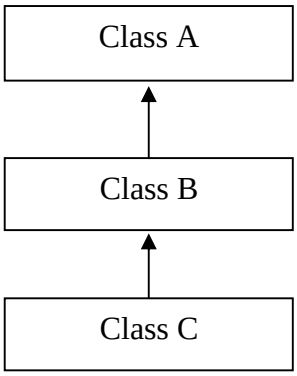


a Base class (super)

it is a sub class (derived)

2. Multilevel Inheritance:

- In Multi level inheritance, a subclass inherits from a class that itself inherits from another class.



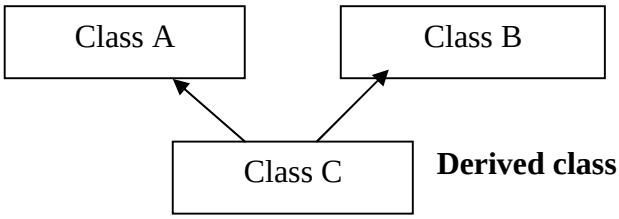
it is a Base class (super) of B

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

derived class(sub) of class B

3. Multiple Inheritance:

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

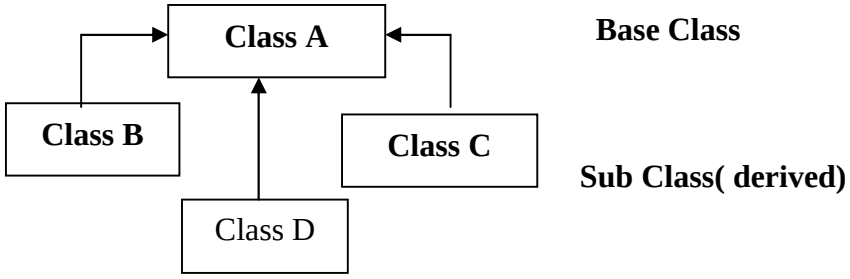


Base class

Derived class

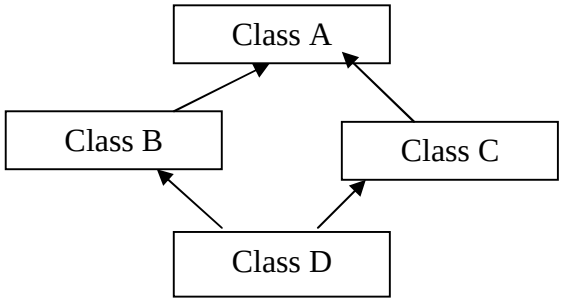
4. Hierarchical Inheritance:

- In hierarchial Inheritance, it's like an inverted tree. So multiple classes inherit from a single base class.



5. Hybrid Inheritance:

- It combines two or more forms of 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**



4 marks Solved Problems :

Q1. **Consider the following declarations and answer the questions given below :**

```
class WORLD
{
int H;
protected :
int S;
public :
void INPUT(int);
void OUTPUT();
};
class COUNTRY : private WORLD
{
int T;
protected :
int U;
public :
void INDATA( int, int)
void OUTDATA();
};
class STATE : public COUNTRY
{
int M;
public :
void DISPLAY (void);};
```

(*i*) Name the base class and derived class of the class COUNTRY.
(*ii*) Name the data member(s) that can be accessed from function DISPLAY().
(*iii*) Name the member function(s), which can be accessed from the objects of class STATE.
(*iv*) Is the member function OUTPUT() accessible by the objects of the class COUNTRY ?
Ans (*i*) Base class : WORLD
Derived class : STATE
(*ii*) M.
(*iii*) DISPLAY(), INDATA() and OUTDATA()
(*iv*) No

Q2. Consider the following declarations and answer the questions given below :

```
class living_being {
char name[20];
protected:
int jaws;
public:
void inputdata(char, int);
void outputdata();
}
class animal : protected living_being {
int tail;
protected:
int legs;
public:
void readdata(int, int);
void writedata();
};
class cow : private animal {
char horn_size;
public:
void fetchdata(char);
void displaydata();
};
```

(*i*) Name the base class and derived class of the class animal.
(*ii*) Name the data member(s) that can be accessed from function displaydata.
(*iii*) Name the data member(s) that can be accessed by an object of cow class.
(*iv*) Is the member function outputdata accessible to the objects of animal class.
Ans (*i*) Base class : living_being
Derived class : cow
(*ii*) horn_size, legs, jaws
(*iii*) fetchdata() and displaydata()
(*iv*) No

Q3. **Consider the following and answer the questions given below :**

```
class MNC
{
char Cname[25]; // Company name
protected :
char Hoffice[25]; // Head office
public :
MNC( );
char Country[25];
void EnterDate( );
void DisplayData( );
};
```

```
class Branch : public MNC
{
long NOE; // Number of employees
char Ctry[25]; // Country
protected:
void Association( );
public :
Branch( );
void Add( );
void Show( );
};
class Outlet : public Branch
{
char State[25];
public :
Outlet();
void Enter();
void Output();};
```

(*i*) Which class's constructor will be called first at the time of declaration of an object of class Outlet?
(*ii*) How many bytes an object belonging to class Outlet require ?
(*iii*) Name the member function(s), which are accessed from the object(s) of class Outlet.
(*iv*) Name the data member(s), which are accessible from the object(s) of class Branch.

Ans (*i*) class MNC
(*ii*) 129
(*iii*) void Enter(), void Output(), void Add(), void Show(), void EnterData(), void DisplayData().
(*iv*) char country[25]

Q4 Consider the following and answer the questions given below :

```
class CEO {
double Turnover;
protected :
int Noofcomp;
public :
CEO( );
void INPUT( );
void OUTPUT( );
};
class Director : public CEO {
int Noofemp;
public :
Director( );
void INDATA();
void OUTDATA( );
protected:
float Funda;
};
class Manager : public Director {
float Expense;
public :
Manager();
void DISPLAY(void);
};
```

(*i*) Which constructor will be called first at the time of declaration of an object of class Manager?
(*ii*) How many bytes will an object belonging to class Manager require ?
(*iii*) Name the member function(s), which are directly accessible from the object(s) of class Manager.
(*iv*) Is the member function OUTPUT() accessible by the objects of the class Director ?
Ans (*i*) CEO()
(*ii*) 16
(*iii*) DISPLAY(), INDATA(), OUTDATA(), INPUT(), OUTPUT()
(*iv*) Yes

4 marks Practice Problems:

Q1 :- 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 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.**

Q2:- Consider the following declarations and answer the questions given below:

```
        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;
        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.

Q3 : 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?

2marks Practice Problems:

1. **What is access specifier ? What is its role ?**
2. **What are the types of inheritance ?**
3. **What is the significance of inheritance ?**
4. **What is the difference between private and public visibility modes?**

DATA FILE HANDLING IN C++

File

- A file is a stream of bytes stored on some secondary storage devices.
- **Text file:** A text file stores information in readable and printable form. Each line of text is terminated with an **EOL** (End of Line) character.
- **Binary file:** A binary file contains information in the non-readable form i.e. in the same format in which it is held in memory.

File Stream

- **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 will use default streams for file input or output. This method is preferred when file is opened in input or output mode only.
    Example : **ofstream file("student.dat"); or ifstream file("student.dat");**
  - **By the open() member function** of the stream. It will preferred when file is opened in various modes i.e ios::in, ios::out, ios::app, ios::ate etc.
    e.g **fstream file;**
    file.open("book.dat", ios::in | ios::out | ios::binary);

File modes:

- **ios::out** It open file in output mode (i.e write mode) and place the file pointer in beginning, if file already exist it will overwrite the file.
- **ios::in** It open file in input mode(read mode) and permit reading from the file.
- **ios::app** It open the file in write mode, and place file pointer at the end of file i.e to add new contents and retains previous contents. If file does not exist it will create a new file.
- **ios::ate** It open the file in write or read mode, and place file pointer at the end of file i.e input/ output operations can performed anywhere 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).

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

Text File functions:

Char I/O :

- **get() –** read a single character from text file and store in a buffer.
  e.g **file.get(ch);**
- **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 **file.getline(s,80);**

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

Binary file functions:

- **read()-** read a block of binary data or reads a fixed number of bytes from the specified stream and store in a buffer.
  Syntax : Stream_object.read((char *)& Object, sizeof(Object));
  e.g file.read((char *)&s, sizeof(s));
- **write() –** write a block of binary data or writes fixed number of bytes from a specific memory location to the specified stream.
  Syntax : Stream_object.write((char *)& Object, sizeof(Object));
  e.g file.write((char *)&s, sizeof(s));

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.

Moving the file pointer in a file for various operations viz modification, deletion , searching etc. Following functions are used:

seekg(): It places the file pointer to the specified position in input mode of file.
e.g file.seekg(p,ios::beg); or file.seekg(-p,ios::end), or file.seekg(p,ios::cur)
i.e to move to **p** byte position from beginning, end or current position.

seekp(): It places the file pointer to the specified position in output mode of file.
e.g file.seekp(p,ios::beg); or file.seekp(-p,ios::end), or file.seekp(p,ios::cur)
i.e to move to **p** byte position from beginning, end or current position.

tellg(): This function returns the current working position of the file pointer in the input mode.
e.g int p=file.tellg();

tellp(): This function returns the current working position of the file pointer in the output mode.
e.f int p=file.tellp();

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
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
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
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
```

2 Marks Questions:

Write a function in a 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'. [CBSE 2012]

```
void DispDorM()
{
        ifstream File("DELHI.TXT")
        char str[80];
        while(File.getline(str,80))
        {
                if(str[0] ='D' || str[0] ='M')
                        cout<<str<<endl;
        }
        File.close(); //Ignore
}
```

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;
}
```

2/3 Marks Practice Questions

1. Write a function in C++ to count the number of uppercase alphabets present in a text file "BOOK.txt"
2. Write a function in C++ to count the number of alphabets present in a text file "BOOK.txt"
3. Write a function in C++ to count the number of digits present in a text file "BOOK.txt"
4. Write a function in C++ to count the number of white spaces present in a text file "BOOK.txt"
5. Write a function in C++ to count the number of vowels present in a text file "BOOK.txt"
6. 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.

General program structure used for operating a Binary File

Program to create a binary file 'student.dat' using structure.

```
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
        ofstream fout;
        char ch;
        fout.open("student.dat", ios::out | ios:: binary);
        clrscr();
        student s;
        if(!fout)
        {
                cout<<"File can't be opened";
                break;
        }
        do
        {
        cout<<"\n enter name of student";
        gets(s);
        cout<<"\n enter persentage";
        cin>>percent;
        fout.write((char *)&s,sizeof(s));  // writing a record in a student.dat file
        cout<<"\n more record y/n";
        cin>>ch;
        }while(ch!='n' || ch!='N');
        fout.close();
}
```

Program to read a binary file 'student.dat' display records on monitor.

```
#include<fstream.h>
struct student
{
char name[15];
float percent;
};
void main()
{
        ifstream fin;
        student s;
        fin.open("student.dat",ios::in | ios:: binary);
        fin.read((char *) &s, sizeof(student));  //read a record from file 'student.dat'
        while(file)
        {
        cout<<s.name;
        cout<<"\n has the percent: "<<s.percent;
        fin.read((char *) &s, sizeof(student));
        }
        fin.close();
}
```

Binary file using Objects and other file operations:

Consider the following class declaration then write c++ function for following file operations viz create_file, read_file, add new records, modify record, delete a record, search for a record.

```
#include<iostream.h>
class student
{
        int rno;
        char name[30];
        int age;
public:
        void input()
        {
                cout<<"\n enter roll no";
                cin>>rno;
                cout<<"\n enter name ";
                gets(name);
                cout<<"\n enter age";
                cin>>age;
        }
        void output()
        {
                cout<< "\n roll no:"<<rno;
                cout<< "\n name :"<<name;
                cout<< "\n age:"<<age;
        }
        int getrno() { return rno;}
};
void create_file()
{
        ofstream fout;
        char ch;
        fout.open("student", ios::out | ios:: binary);
```

```
        clrscr();
        student s;
        if(!fout)
        {cout<<“File can’t be opened”;
                break;
        }
        do
        {
        s.input();
        cout<<”\n more record y/n”;
        cin>>ch;
        }while(ch!=’n’ || ch!=’N’);
        fout.close();
}
void read_file()
{
        ifstream fin;
        student s;
        fin.open(“student.dat”,ios::in | ios:: binary);
        fin.read((char *) &s,sizeof(student));
        while(file)
        {
        s.output();
        cout<< “\n”;
        fin.read((char *) & s,sizeof(student));
        }
        fin.close();
}
void modify_record()
{       student s;
        fstream file;
        file.open(“student.dat”,ios::in|ios::out|ios::ate|ios::binary);
        int r,pos=-1;
        cout<<”\n enter the rollo no of student whom data to be modified”;
        cin>>r;
        file.read((char *)&s,sizeof(s));
        while(file)
        {
                if (r==s.getrno())
                {
                cout<<”\n record is “;
                s.output();
                pos =file.tellg()-size(s);
                break;
                }
        file.read((char *)&s,sizeof(s));
        }
        if(pos>-1)
        {
        cout<< “\n enter new     record”;
        s.input();
        file.seekp(pos,ios::beg);
        file.write((char *)&s,sizeof(s));
        cout<< “\n record modified successfully”;
        }

        else
        cout<< “\n record not exist”;
}
```

```
void delete_record()
{
fstream file("student.dat", ios::in|ios::binary);
fstream newfile("newstu.dat",ios::out|ios::binary);
student s;
cout<<"\n enter the rollno no of student whom record to be deleted";
cin>>r;
file.read((char *)&s,sizeof(s));
while(file)
        {
                if (r!=s.getrno())
                {
                newfile.write((char *)&s,sizeof(s));
                }
        file.read((char *)&s,sizeof(s));
        }
file.close();
newfile.close();
}
void search_record()
{
student s;
        fstream file;
        file.open("student.dat",ios::in|os::binary);
        int r,flag=-1;
        cout<<"\n enter the rollo no of student whom record to be searched";
        cin>>r;
        file.read((char *)&s,sizeof(s));
        while(file)
        {
                if (r==s.getrno())
                {
                cout<<"\n record is ";
                s.output();
                flag=1;
                break;
                }
        file.read((char *)&s,sizeof(s));
        }
        if(flag==1)
        cout<< "\n search successfull";
        else
        cout<< "\n search unsuccessfull";
        file.close();
}
```

1 Mark Questions

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

```
class stock
{
        int Ino, Qty; Char Item[20];
public:
        void Enter() { cin>>Ino; gets(Item); cin>>Qty;}
        void issue(int Q) { Qty+=0;}
        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= = ss.GetIno())
            {
                s.Purchase(PQty);
                _____    // statement 1
                _____    // statement 2
                Success++;
            }
        }
    if (success = =1)
        cout<< "Purchase Updated"<<endl;
    else
        cout<< "Wrong Item No"<<endl;
    File.close() ;
    }
```

Ans.1. i) Statement 1 to position the file pointer to the appropriate place so that the data updation is done for the required item.

File.seekp(File.tellg()-sizeof(stock);
OR
File.seekp(-sizeof(stock),ios::cur);

ii) Staement 2 to perform write operation so that the updation is done in the binary file.
File.write((char *)&s, sizeof(s)); OR File.write((char *)&s, sizeof(stock));

3 Marks Question

2. Write a function in c++ to search for details (Phoneno and Calls) of those Phones which have more than 800 calls from binary file "phones.dat". Assuming that this binary file contains records/ objects of class Phone, which is defined below.

CBSE 2012

```
class Phone
{
    Char Phoneno[10]; int  Calls;
public:
    void Get() {gets(Phoneno); cin>>Calls;}
    void Billing() { cout<<Phoneno<< "#"<<Calls<<endl;}
    int GetCalls() {return Calls;}
};
```

Ans 2 :

```
void Search()
{
    Phone P;
    fstream fin;
    fin.open( "Phone.dat", ios::binary| ios::in);
    while(fin.read((char *)&P, sizeof(P)))
    {
        if(p.GetCalls() >800)
            p.Billing();
    }
    Fin.close(); //ignore
}};
```

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;}
};
```

Ans.3. void searchbook(int bookno)

```
{ifstream ifile("BOOK.DAT",ios::in|ios::binary);
 if(!ifile)
        {cout<<"could not open BOOK.DAT file"; exit(-1);}
else
        {BOOK b; int found=0;

                while(ifile.read((char *)&b, sizeof(b)))
                {if(b.RBno()==bookno)
                        {b.Display(); found=1; break;}
                }
        if(! found)
                cout<<"record is not found ";
        ifile.close();
        }
}
```

4. **Given a binary file PHONE.DAT, containing records of the following class type**

```
 class Phonlist
{
char name[20];
char address[30];
char areacode[5];
char Phoneno[15];
public:
void Register()
void Show();
void CheckCode(char AC[])
{return(strcmp(areacode,AC);
}};
```

Write a function TRANSFER( ) in C++, that would copy all those records which are having areacode as "DEL" from PHONE.DAT to PHONBACK.DAT.

Ans 4. void TRANSFER()

```
{
        fstream File1,File2;
        Phonelist P;
        File1.open("PHONE.DAT", ios::binary|ios::in);
        File2.open("PHONEBACK.DAT", ios::binary|ios::OUT)
        while(File1.read((char *)&P, sizeof(P)))
        {               if( p.CheckCode( "DEL"))
                        File2.write((char *)&P,sizeof(P));      }
       File1.close();
       File2.close();
}
```

POINTERS

- Pointer is a variable that holds a memory address of another variable of same type.
- It supports dynamic allocation routines.
- It can improve the efficiency of certain routines.

C++ Memory Map :

- Program Code : It holds the compiled code of the program.
- Global Variables : They remain in the memory as long as program continues.
- Stack : 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 : It is a region of free memory from which chunks of memory are allocated via DMA functions.

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.
e.g. int a; // This will allocate 2 bytes for a during compilation.

Dynamic Memory Allocation : The amount of memory to be allocated is not known beforehand rather it is required to allocated as and when required during runtime, it is referred to as dynamic memory allocation.
C++ offers two operator for DMA – **new and delete**.
e.g int x =new int; float y= new float; // dynamic allocation
delete x; delete y; //dynamic deallocation

Free Store : It is a pool of unallocated heap memory given to a program that is used by the program for dynamic memory allocation during execution.

Declaration and Initialization of Pointers :

```
Datatype *variable_name;
```

Syntax : Datatype *variable_name;

```
Int *p;        float *p1;      char *c;
```

Eg. Int *p; float *p1; char *c;

Two special unary operator * and & are used with pointers. The & is a unary operator that returns the memory address of its operand.

Eg. Int a = 10; int *p; p = &a;

Pointer arithmetic:

Two arithmetic operations, addition and subtraction, may be performed on pointers.
When you add 1 to a pointer, you are actually adding the size of whatever the pointer is pointing at. That is, each time a pointer is incremented by 1, it points to the memory location of the next element of its base type.
e.g. int *p; P++;
If current address of p is 1000, then p++ statement will increase p to 1002, not 1001.

If *c is char pointer and *p is integer pointer then

Char pointer	C	c+1	c+2	c+3	c+4	c+5	c+6	c+7
Address	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**
Int pointer	p		p+1		p+2		p+3	

Adding 1 to a pointer actually adds the size of pointer's base type.
Base address : A pointer holds the address of the very first byte of the memory location where it is pointing to. **The address of the first byte is known as BASE ADDRESS.**

Dynamic Allocation Operators :

C++ dynamic allocation allocate memory from the free store/heap/pool, the pool of unallocated heap memory provided to the program. C++ defines two unary operators **new** and **delete** that perform the task of allocating and freeing memory during runtime.

Creating Dynamic Array :

Syntax : pointer-variable = new data-type [size];

e.g. int * array = new int[10];

Not array[0] will refer to the first element of array, array[1] will refer to the second element.

No initializes can be specified for arrays.

All array sizes must be supplied when new is used for array creation.

Two dimensional array :

```
int *arr, r, c;
r = 5; c = 5;
arr = new int [r * c];
Now to read the element of array, you can use the following loops :
For (int i = 0; i < r; i++)
{
        cout << "\n Enter element in row " << i + 1 << " : ";
        For (int j=0; j < c; j++)
                cin >> arr [ i * c + j];
}
```

Memory released with delete as below:

Syntax for simple variable : delete pointer-variable; eg. delete p;	For array : delete [size] pointer variable; Eg. delete [ ] arr;

Pointers and Arrays :

C++ treats the name of an array as constant pointer which contains base address i.e address of first location of array. Therefore Pointer variables are efficiently used with arrays for declaration as well as accessing elements of arrays, because array is continuous block of same memory locations and therefore pointer arithmetic help to traverse in the array easily.

```
void main()
  {
        int *m;
        int marks[10] ={ 50,60,70,80,90,80,80,85,75,95};
        m = marks; // address of first location of array  or we can write it as m=&marks[0]
        for(int i=0;i<10;i++)
        cout<< *m++;
          // or
        m = marks; // address of first location of array  or we can write it as m=&marks[0]
        for(int i=0;i<10;i++)
        cout<< *(m+i);
  }
```

Array of Pointers :

To declare an array holding 10 int pointers –
int * ip[10];
That would be allocated for 10 pointers that can point to integers.
Now each of the pointers, the elements of pointer array, may be initialized. To assign the address of an integer variable phy to the forth element of the pointer array, we have to write ip[3] = & phy;
Now with *ip[3], we can find the value of phy. int *ip[5];

Index	0	1	2	3	4
address	1000	1002	1004	1006	1008

int a = 12, b = 23, c = 34, d = 45, e = 56;

Variable	a	b	c	d	e
Value	12	23	34	45	56
address	1050	1065	2001	2450	2725

ip[0] = &a; ip[1] = &b; ip[2] = &c; ip[3] = &d; ip[4] = &e;

Index	ip[0]	ip[1]	ip[2]	ip[3]	ip[4]
Array ip value	1050	1065	2001	2450	2725
address	1000	1002	1004	1006	1008

ip is now a pointer pointing to its first element of ip. Thus ip is
equal to address of ip[0], i.e. 1000
*ip (the value of ip[0]) = 1050
* (* ip) = the value of *ip = 12
* * (ip+3) = * * (1006) = * (2450) = 45

Pointers and Strings :

Pointer is very useful to handle the character array also. E.g :

```
void main()
{       char str[] = “computer”;
        char *cp;
        cp=str;
        cout<<str ; //display string
        cout<<cp; // display string
        for (cp =str; *cp != ‘\0’; cp++)   // display character by character by character
                cout << ”--“<<*cp;
        // arithmetic
        str++; // not allowed because str is an array and array name is constant pointer
       cp++; // allowed because pointer is a variable
       cout<<cp;}
```

Output :

Computer
Computer
--c--o--m--p--u--t--e—r
omputer

An array of char pointers is very useful for storing strings in memory. Char *subject[] = { “Chemistry”, “Phycics”, “Maths”, “CS”, “English” };

In the above given declaration subject[] is an array of char pointers whose element pointers contain base addresses of respective names. That is, the element pointer subject[0] stores the base address of string “Chemistry”, the element pointer subject[1] stores the above address of string “Physics” and so forth.

An array of pointers makes more efficient use of available memory by consuming lesser number of bytes to store the string.

An array of pointers makes the manipulation of the strings much easier. One can easily exchange the positions of strings in the array using pointers without actually touching their memory locations.

Pointers and CONST :

A constant pointer means that the pointer in consideration will always point to the same address. Its address can not be modified.

A pointer to a constant refers to a pointer which is pointing to a symbolic constant. Look the following example :

```
int m = 20;                     // integer m declaration
int *p = &m;                    // pointer p to an integer m
++ (*p);                        // ok : increments int pointer p
int * const c = &n;             // a const pointer c to an intger n
++ (* c);                       // ok : increments int pointer c i.e. its contents
++ c;                           // wrong : pointer c is const – address can’t be modified
const int cn = 10;              // a const integer cn
const int *pc = &cn;            // a pointer to a const int
++ (* pc);                      // wrong : int * pc is const – contents can’t be modified
++ pc;                          // ok : increments pointer pc
const int * const cc = *k;      // a const pointer to a const integer
++ (* cc);                      // wrong : int *cc is const
++ cc;                          // wrong : pointer cc is const
```

Pointers and Functions :

A function may be invoked in one of two ways :

1. call by value
2. call by reference

The second method call by reference can be used in two ways :

1. by passing the references
2. by passing the pointers

Reference is an alias name for a variable. For ex : int m = 23;

```
int &n = m;
int *p;
p = &m;
```

Then the value of m i.e. 23 is printed in the following ways : cout << m; // using variable name

```
cout << n;      // using reference name
cout << *p;     // using the pointer
```

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 the original 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
After swapping value of a : 6 and b : 5

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 :";          cint >> 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)
}
```

Dynamic structures :

The new operator can be used to create dynamic structures also i.e. the structures for which the memory is dynamically allocated.

struct-pointer = new struct-type;

student *stu;

stu = new Student;

A dynamic structure can be released using the deallocation operator delete as shown below :

delete stu;

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 called function 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.

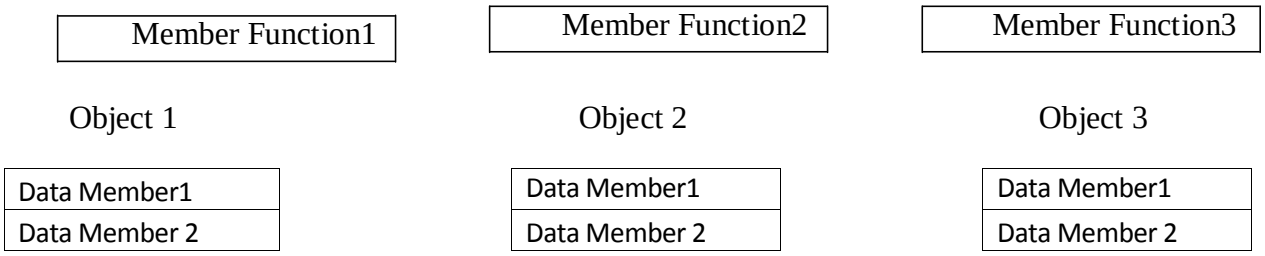
```
student stud[5], *sp;
---
sp = stud;              // sp points to the first element (stud[0])of stud
sp++;                   // sp points to the second element (stud[1]) of stud sp + = 2;

// sp points to the fourth element (stud[3]) of stud sp--;                    // sp

points to the third element (stud[2]) of stud
```

You can even make a pointer point to a data member of an object. Two points should be considered :

1. A Pointer can point to only public members of a class.
2. The data type of the pointer must be the same as that of the data member it points to.

this Pointer :

In class, the member functions are created and placed in the memory space only once. That is only one copy of functions is used by all objects of the class.
Therefore if only one instance of a member function exists, how does it come to know which object's data member is to be manipulated?



For the above figure, if Member Function2 is capable of changing the value of Data Member3 and we want to change the value of Data Member3 of Object3. How would the Member Function2 come to know which Object's Data Member3 is to be changed?
To overcome this problem this pointer is used.
When a member function is called, it is automatically passed an implicit argument that is a pointer to the object that invoked the function. This pointer is called This.
That is if ojbect3 is invoking member function2, then an implicit argument is passed to member function2 that points to object3 i.e. this pointer now points to object3.
The friend functions are not members of a class and, therefore, are not passed a this pointer. The static member functions do not have a this pointer.

Solved Questions

Q. 1 How is *p different from **p ?

Ans : *p means, it is a pointer pointing to a memory location storing a value in it. But **p means, it is a pointer pointing to another pointer which in turn points to a memory location storing a value in it.

Q. 2 How is &p different from *p ?

Ans : &p gives us the address of variable p and *p. dereferences p and gives us the value stored in memory location pointed to by p.

Q. 3 Find the error in following code segment :

```
Float **p1, p2;
    P2 = &p1;
```

Ans : In code segment, p1 is pointer to pointer, it means it can store the address of another pointer variable, whereas p2 is a simple pointer that can store the address of a normal variable. So here the statement p2 = &p1 has error.

Q. 4 What will be the output of the following code segment ?

```
char C1 = 'A';
char C2 = 'D';
char *i, *j;
i = &C1;
j = &C2;
*i = j;
cout << C1;
```

Ans : It will print A.

Q. 5 How does C++ organize memory when a program is run ?

Ans : Once a program is compiled, C++ creates four logically distinct regions of memory :

(i) area to hold the compiled program code
(ii) area to hold global variables
(iii) the stack area to hold the return addresses of function calls, arguments passed to the functions, local variables for functions, and the current state of the CPU.
(iv) The heap area from which the memory is dynamically allocated to the program.

Q. 6 Identify and explain the error(s) in the following code segment :

```
float a[] = { 11.02, 12.13, 19.11, 17.41};
float *j, *k;
j = a;
k = a + 4;
j = j * 2;
k = k / 2;
cout << " *j = " << *j << ", *k = " << *k << "\n";
```

Ans : The erroneous statements in the code are :

```
j = j * 2;
k = k / 2;
```

Because multiplication and division operations cannot be performed on pointer and j and k are pointers.

Q. 13 How does the functioning of a function differ when

(i) an object is passed by value ? (ii) an object is passed by reference ?

Ans : (i) When an object is passed by value, the called function creates its own copy of theobject by just copying the contents of the passed object. It invokes the object's copy constructor to create its copy of the object. However, the called function destroys its copy of the object by calling the destructor function of the object upon its termination.

(i) When an object is passed by reference, the called function does not create its own copy of the passed object. Rather it refers to the original object using its reference or alias name. Therefore, neither constructor nor destructor function of the object is invoked in such a case.

2 MARKS PRACTICE QUESTIONS

1. Differentiate between static and dynamic allocation of memory.

2. 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;
    }
```

3. 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;
    }
```

4. Identify the syntax error(s), if any, in the following program. Also give reason for errors.

```
void main()
{const int i = 20;
    const int * const ptr = &i;
    (*ptr++; int j= 15; ptr
    = &j;  }
```

5. What is 'this' pointer ? What is its significance ?

6. What will be the output of following program ?

```
#include<iostream.h>
void main()
{
    char name1[] = "ankur"; char
    name2[] = "ankur"; if (name1 !=
    name2)
            cout << "\n both the strings are not equal";
     else
          cout << "\n the strings are equal";              }
```

7. Give and explain the output of the following code :

```
    void junk (int, int *);
int main()                    {
    int i = 6, j = -4;
    junk (i, &j);
    cout << "i = " << i << ", j = " << j << "\n";
    return 0;                 }
void junk(int a, int *b)
{
    a = a* a;
        *b = *b * *b;                    }
```

UNIT-2 DATA STRUCTURES

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. For example, Stacks are used in function call during execution of a program, while B-trees are particularly well-suited for implementation of databases. 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 5^{th} element of the array which is 9 and b[5] denotes 6^{th} 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).

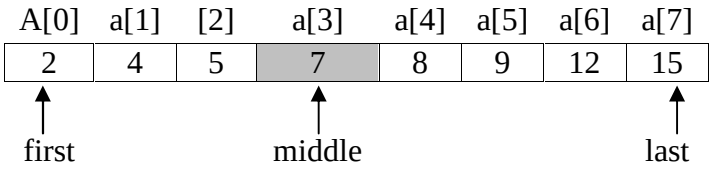
```
#include<iostream.h>
int binary_search(int a[ ], int size, int item)
{
int first=0,last=size-1,middle;
while(first<=last)
        {
        middle=(first+last)/2;
```

```
        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



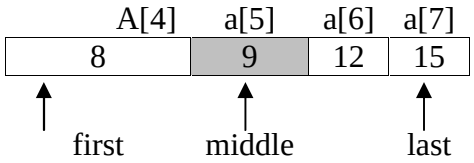
First=0, last=7

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



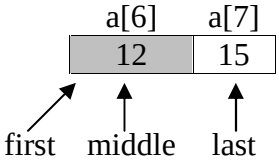
first=4, last=7

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";
        }
 else
        cout<<"\n Given item is not found in the array";
n--;
}
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).

```
#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 i^th 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]<<", ";
}
```

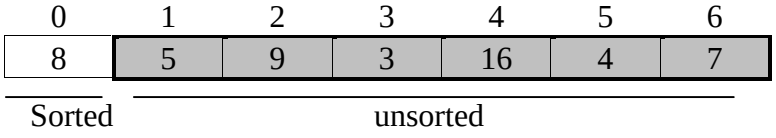
Output is
Original array is
8, 5, 9, 3, 16, 4, 7
The sorted array is
3, 4, 5, 7, 8, 9, 16

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:

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

Sorted unsorted

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:

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

Sorted unsorted

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:

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

Sorted unsorted

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:

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

Sorted unsorted

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

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

Sorted unsorted

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

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

Sorted

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.

```
#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]<<", ";
}
```

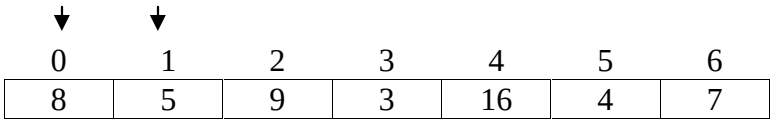
Output is
Original array is
8, 5, 9, 3, 16, 4, 7
The sorted array is
3, 4, 5, 7, 8, 9, 16

Bubble Sort

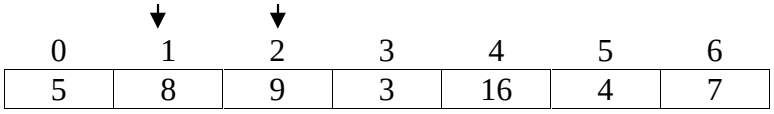
Bubble sort compares a[i] with a[i+1] for all i=0..n-2, if a[i] and a[i+1] are not in ascending order then exchange a[i] with a[i+1] immediately. After each iteration all elements which are not at their proper position move at least one position towards their right place in the array. The process continues 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

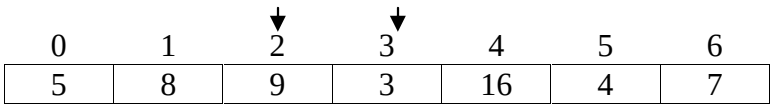
Original array



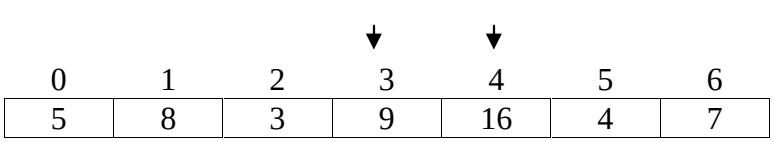
Iteration1: The element a[0] i.e. 8 is compared with a[1] i.e. 5, since 8>5 therefore exchange 8 with 5.



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]



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.

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

As C, C++ supports n dimensional arrays along the row, the address calculation formula can be generalized for n dimensional array as:
For 3 dimentional array A[m][n][p], find address of a[i][j][k]:
Address of a[i][j][k] = base address + ((I * n + j) * p + k) * sizeof(type)

For 4 dimentional array A[m][n][p][q], find address of a[i][j][k][l]:
Address of a[i][j][k][l] = base address + (((I * n + j) * p + k) * p + l) * sizeof(type)

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+215 =5215

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= 4000-55*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

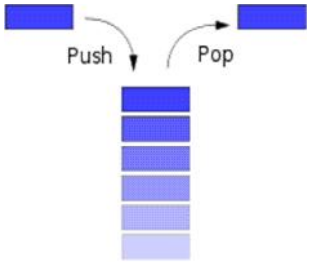
1. Write a function in C++ which accepts an integer array and its size as arguments and replaces elements having even values with its half and elements having odd values with twice its value
2. Write a function in C++ which accepts an integer array and its size as argument and exchanges the value of 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

STACKS, QUEUES 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.

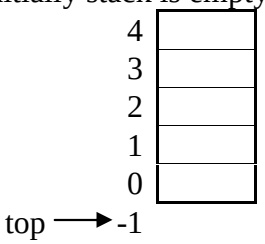


Stack using array

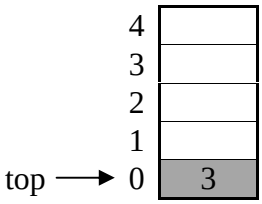
```
#include<iostream.h>
const int size=5
class stack
{int a[size];     //array a can store maximum 5 item of type int of the stack
int top;          //top will point to the last item pushed onto the stack
public:
stack(){top=-1;}        //constructor to create an empty stack, top=-1 indicate that no 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.

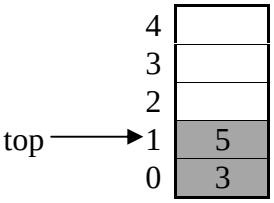
Initially stack is empty



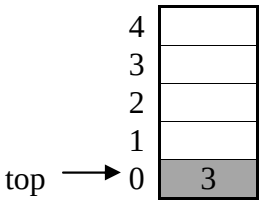
stack after s1.push(3)



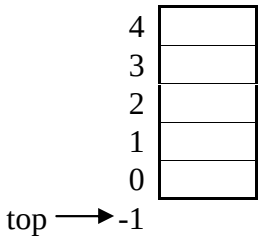
stack after s1.push(5)



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



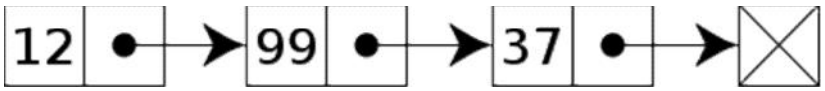
After second s1.pop() statement, the item 3 is removed from stack and top moves from 0 to -1 which indicates that now stack is empty.



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.

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
```

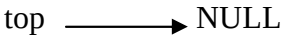
```
{
int item; //data that will be stored in each node
node * next; //pointer which contains address of another node
}; //node is a self referential structure which contains reference of another object type node

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;
}
```

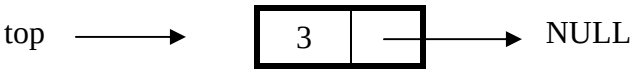
Output is

7
5
3
Stack is empty 0

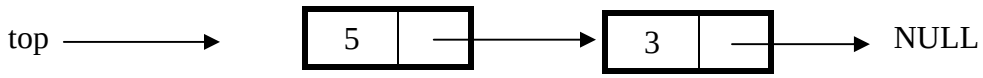
In the above program the statement *stack s1;* invokes the constructor stack() which create an empty stack object s1 and initialize top with NULL.



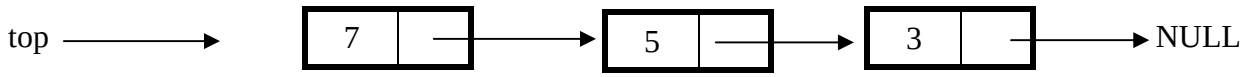
After statement s1.push(3) the stack become



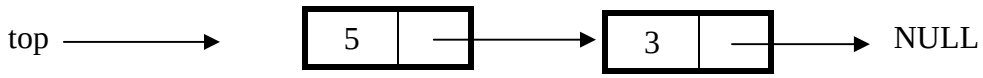
After statement s1.push(5) the stack become



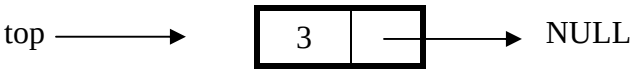
After statement s1.push(7) the stack become



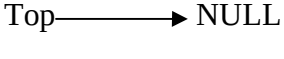
After the first s1.pop() statement the node currently pointed by top (i.e. node containing 7) is deleted from the stack, after deletion the status of stack is



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

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.

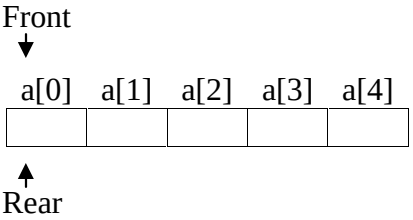
Linear Queue implementation using Array

```
#include<iostream.h>
const int size=5;
class queue
{int front , rear;
int a[size];
public:
queue(){front=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;
}
```

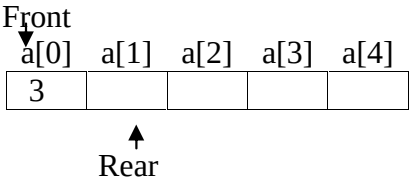
Output is

3
5
7
Queue is empty 0

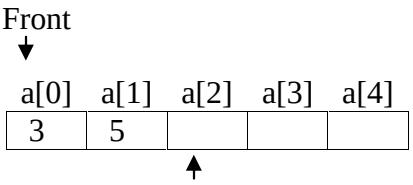
In the above program the statement **queue q1** creates an empty queue q1.



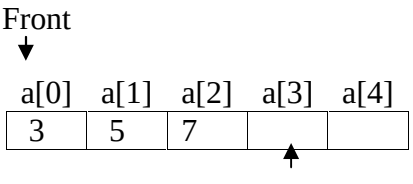
After execution of the statement **q1.addQ(3),** status of queue is



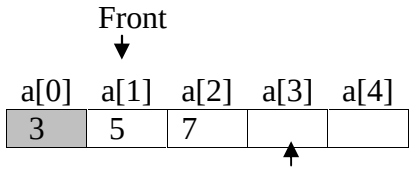
After execution of the statement **q1.addQ(5),** status of queue is



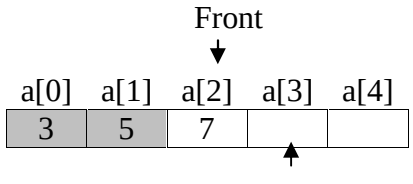
After execution of the statement **q1.addQ(7),** status of queue is



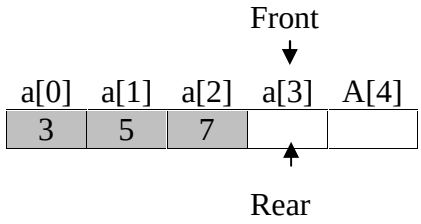
After execution of the first cout<<**q1.delQ()** statement**,** 3 is deleted from queue status of queue is



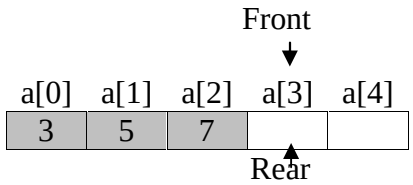
After execution of the second cout<<**q1.delQ()** statement**, 5** is deleted from the queue status of queue is



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



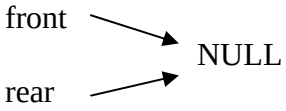
Note that since rear and front moves only in one direction therefore once the rear cross the last element of the array(i.e. rear==size) then even after deleting some element of queue the free spaces available in queue cannot be allocated again and the function delQ() display error message "queue is full".

Queue using linked list

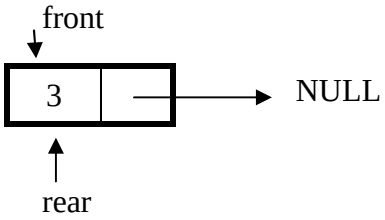
```
#include<iostream.h>
struct node{
int item;
node *next;};
class queue
{node *front, *rear;
public:
queue() {front=NULL; rear=NULL;}//constructor to create empty queue
void addQ(int item);
int delQ();};
void queue::addQ(int item)
{node * t=new node;
if(t==NULL)
        cout<<"memory not available,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

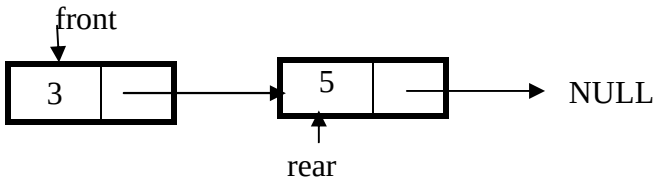
In the above program the statement **queue q1;** invokes the constructor queue() which create an empty queue object q1 and initialize front and rear with NULL.



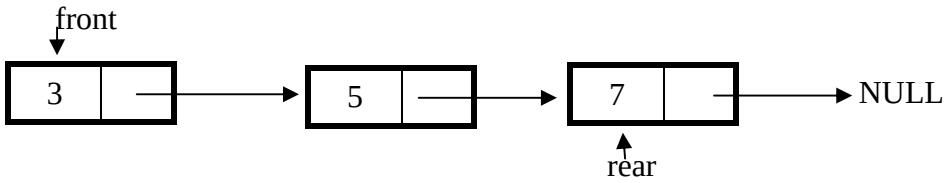
After statement q1.addQ(3) the stack become



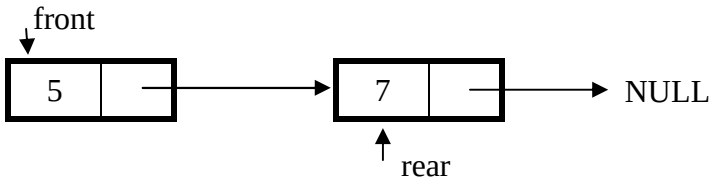
After statement q1.addQ(5) the stack become



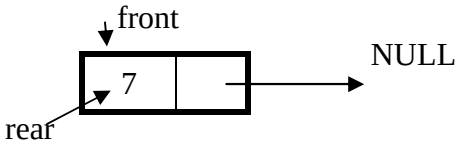
After statement q1.addQ(7) the stack become



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

Front=rear= NULL ;

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
```

2,3 & 4 Marks Practice Questions

1. Convert the following infix expressions to postfix expressions using stack 2
   1. A + (B * C) ^ D – (E / F – G)
   2. A * B / C * D ^ E * G / H
   3. ((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
   2. E= 80,35,20,-,25,5,+,-,*
   3. E= 30,5,2,^,12,6,/,+,-
   4. E=15, 3, 2, +, /, 7, + 2, *

3. 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. 3

4 Define functions in C++ to perform a PUSH and POP operation in a dynamically allocated stack considering the following : 4

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

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

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

UNIT-3 DATABASE AND SQL

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

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.

Relation: Relation (collection of rows and columns) generally refers to an active entity on which we can perform various operations.

Database: Collection of logically related data along with its description is termed as database.

Tuple: A row in a relation is called a tuple.

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.

Primary Key: Primary key is a key that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Foreign Key: Foreign Key is a key that is defined as a primary key in some other relation. This key is used to enforce referential integrity in RDBMS.

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.

DBA: Data Base Administrator is a person (manager) that is responsible for defining the data base schema, setting security features in database, ensuring proper functioning of the data bases etc.

Relational Algebra: Relation algebra is a set operation such as select, project, union, & Cartesian product etc.

Select operation : Yields set of set of rows depending upon certain condition. Mathematically it is denoted as (σ)

e.g $\sigma_{rollno>10}$(student) -means show those rows of student table whose roll no.'s are >10.

Project Operation **:** yields set of columns as result which are specified. Mathematically it is denoted as π.

e.g. $\pi_{rollno,name}$(student) - means show only rollno and name column only.

Union Operation : Two relation are said to be union compatible if their degree and column are same.

e.g Relation A

	X	Y	Z
Relation A	x1	y1	z1
	x2	y2	z2

	X	Y	Z
Relation B	x1	y1	z1
	x3	y3	z3

Resultant Relation A∪B=

X	Y	Z
x1	y1	z1
x2	y2	z2
x3	y3	z3

Cartesian product: Cartesian Product of two relation A and B gives resultant relation whose no. of column are sum of degrees of two relation and no. of rows are product of cardinality of two relations.

e.g Relation A

Relation A

X	Y	Z
x1	y1	z1
x2	y2	z2

Relation B

U	V
u1	v1
u2	v2

Resultant Relation A×B =

X	Y	Z	U	V
x1	y1	z1	u1	v1
x1	y1	z1	u2	v2
x2	y2	z2	u1	v1
x2	y2	z2	u2	v2

Structured Query Language

SQL is a non procedural language that is used to create, manipulate and process the databases(relations).

Characteristics of SQL

1. It is very easy to learn and use.
2. Large volume of databases can be handled quite easily.
3. It is non procedural language. It means that we do not need to specify the procedures to accomplish a task but just to give a command to perform the activity.
4. SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

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. **Data Control Language:**

   This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc are available in DCL.

4. **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 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) CHAR
3) VARCHAR / VARCHAR2
4) DATE
5) LONG
6) RAW/LONG RAW

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. LONG

This data type is used to store variable length strings of upto 2 GB size.

e.g description long;

6. RAW/LONG RAW

To store binary data (images/pictures/animation/clips etc.) RAW or LONG RAW data type is used. A column LONG RAW type can hold upto 2 GB of binary data.

e.g image raw(2000);

SQL Commands

CREATE TABLE Command:

Create table command is used to create a table in SQL. It is a DDL type of command.

Syntax :

CREATE TABLE <table name>
(<column name> <data types>[(size)] [,<column name> <data types>[(size)]….);
e.g.
Create table student
(rollno number(2), name
varchar2(20), dob date);

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.They are used to ensure integrity of a relation, hence named as **integrity constraints**.

1. NOT NULL
2. UNIQUE

3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

i. **Not Null constraint** : It ensures that the column cannot contain a NULL value.

ii. **Unique constraint** : A candidate key is a combination of one or more columns, the value of which uniquely identifies each row of a table.

iii. **Primary Key** : It ensures two things : (i) Unique identification of each row in the table. (ii) No column that is part of the Primary Key constraint can contain a NULL value.

iv. **Foreign Key** : The foreign key designates a column or combination of columns as a foreign key and establishes its relationship with a primary key in different table.

```
Create table Fee
(RollNo number(2) Foreign key (Rollno) references Student (Rollno),
Name varchar2(20) Not null, Amount
number(4), Fee_Date date);
```

v. **Check Constraint :** Sometimes we may require that values in some of the columns of our table are to be within a certain range or they must satisfy certain conditions.

Example:

```
Create table Employee
(EmpNo number(4) Primary Key,
Name varchar2(20) Not Null,
Salary number(6,2) check (salary > 0),
DeptNo number(3)
);
```

Data Manipulation in SQL

DML Commands are as under:

SELECT - Used for making queries
INSERT - Used for adding new row or record into table
UPDATE- used for modification in existing data in a table
DELETE – used for deletion of records.

INSERT Statement

To insert a new 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

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.
Other operators :

4. Range check – between low and high
5. List check – in
6. Pattern check – like , not like (% and _ 'under score' is used)

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.
e.g. SELECT name, class FROM student;

The above command displays only name and class attributes from student table.

Projection: This is the capability of SQL to return only specific attributes in the relation. Use of **where clause** is required when specific tuples are to be fetched or manipulated.

e.g 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 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

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.

SQL Functions :

SQL supports functions which can be used to compute and select numeric, character and date columns of a relations. These functions can be applied on a group of rows. The rows are grouped on a common value of a column in the table. These functions return only one value for a group and therefore, they are called aggregate or group functions.

1. **SUM() :**
   It returns the sum of values of numeric type of a column.
   Eg. Select sum(salary) from employee;
2. **AVG() :**
   It returns the average of values of numeric type of a column.

Eg. Select avg(salary) from employee;

3. **MIN() :**
It returns the minimum of the values of a column of a given relation.

Eg. Select min(salary) from employee;

4. **MAX() :**
It returns the maximum of the values of a column of a given relation.

Eg. Select max(salary) from employee;

5. **COUNT():**
It returns the number of rows in a relation.

Eg. Select count(*) from employee;

Group By Clause :
The rows of a table can be grouped together based on a common value by using the Group By clause of SQL in a select statement.
Syntax :
SELECT <attribute name>, <attribute name> ---- [functions]
FROM <relation name>
GROUP BY <group by column>;
Eg. Select age, count (rollno)
From students Group by age;
Output :

Age	Count(rollno)
15	2
14.5	2
14	5

Group-By-Having Clause :
It is used to apply some condition to the Group By clause.
Eg.
Select class
From students Group by class
Having count(*) > 5;

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, UPDATE 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

1&2 mark questions

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

Ans:. 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.

2 marks Practice 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 category.

Database and SQL : 6 marks questions

1. Write SQL Command for (a) to (d) and output of (g)

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 I 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 SUBJECT) from GRADUATE;

Sol :

(b) SELECT NAME from GRADUATE where DIV = 'I' order by NAME;

(c) SELECT NAME,STIPEND,SUBJECT, STIPEND*12 from GRADUATE;

(d) SELECT SUBJECT,COUNT(*) from GRADUATE group by SUBJECT having SUBJECT='PHYISCS' or SUBJECT='COMPUTER SC';

(e) INSERT INTO GRADUATE values(11,'KAJOL',300,'COMPUTER SC',75,1);

(f) i) 63
ii) 800
iii) 475
iv) 4

2. Consider the following tables Sender and Recipient. Write SQL commands for the statements (i) to (iv) and give the outputs for SQL queries (v) to (viii).

Sender

SenderID	SenderName	SenderAddress	City
ND01	R Jain	2, ABC Appts	New Delhi
MU02	H Sinha	12, Newtown	Mumbai
MU15	S Jha	27/A, Park Street	Mumbai
ND50	T Prasad	122 – K, SDA	New Delhi

Recipient

RecID	SenderID	RecName	RecAddress	RecCity
KO05	ND01	R Bajpayee	5, Central Avenue	Kolkata
ND08	MU02	S Mahajan	116, A-Vihar	New Delhi
MU19	ND01	H Singh	2A, Andheri East	Mumbai
MU32	MU15	P K Swamy	B5, C S Terminus	Mumbai
ND48	ND50	S Tripathi	13, B I D, Mayur Vihar	New Delhi

(i) To display the names of all Senders from Mumbai

Ans. SELECT sendername from Sender where sendercity='Mumbai';

(ii) To display the RecIC, Sendername, SenderAddress, RecName, RecAddress for every Recipient.

Ans. Select R.RecIC, S.Sendername, S.SenderAddress, R.RecName, R.RecAddress from Sender S, Recepient R where S.SenderID=R.SenderID;

(iii) To display Recipient details in ascending order of RecName

Ans. SELECT * from Recipent ORDER By RecName;

(iv) To display number of Recipients from each city

Ans. SELECT COUNT(*) from Recipient Group By RecCity;

(v) SELECT DISTINCT SenderCity from Sender; Ans.

SenderCity
Mumbai
New Delhi

(vi) SELECT A.SenderName, B.RecName
From Sender A, Recipient B
Where A.SenderID = B.SenderID AND B.RecCity ='Mumbai';

Ans.

A.SenderName	B.RecName
R Jain	H Singh
S Jha	P K Swamy

(vii) SELECT RecName, RecAddress
From Recipient Where RecCity NOT IN ('Mumbai', 'Kolkata');

Ans. RecName RecAddress
S Mahajan 116, A Vihar
S Tripathi 13, BID, Mayur Vihar

(viii) SELECT RecID, RecName
FROM Recipent Where SenderID='MU02' or SenderID='ND50';

Ans. RecID RecName ND08
S Mahajan ND48 S
Tripathi

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) Add a new column named 'Marks'.
(f) Assign a value 200 for Marks for all those who are getting grade 'B' or grade 'A' in both Game1 and Game2.

Ans : a) SELECT Name from SPORTS where grade='C' or Grade2='C';
b) SELECT Count(*) from SPORTS where grade='A';
c) SELECT name from SPORTS where game1 = game2;
d) SELECT game,game2 from SPORTS where name like 'A%';
e) ALTER TABLE SPORTS add (marks int(4));
f) UPDATE SPORTS set marks=200 where grade='A';

4. Consider the following tables Stationary and Consumer. Write SQL commands for the statement (i) to (iv) and output for SQL queries (v) to (viii):

Table: Stationary

S_ID	StationaryName	Company	Price
DP01	Dot Pen	ABC	10
PL02	Pencil	XYZ	6
ER05	Eraser	XYZ	7
PL01	Pencil	CAM	5
GP02	Gel Pen	ABC	15

Table: Consumer

C_ID	ConsumerName	Address	S_ID
01	Good Learner	Delhi	PL01
06	Write Well	Mumbai	GP02
12	Topper	Delhi	DP01
15	Write & Draw	Delhi	PL02
16	Motivation	Banglore	PL01

(i) To display the details of those consumers whose Address is Delhi.
(ii) To display the details of Stationary whose Price is in the range of 8 to 15. (Both Value included)
(iii) To display the ConsumerName, Address from Table Consumer, and Company and Price from table Stationary, with their corresponding matching S_ID.
(iv) To increase the Price of all stationary by 2.
(v) SELECT DISTINCT Address FROM Consumer;
(vi) SELECT Company, MAX(Price), MIN(Price), COUNT(*) from Stationary GROUP BY Company;
(vii) SELECT Consumer.ConsumerName, Stationary.StationaryName, Stationary.Price FROM Strionary, Consumer WHERE Consumer.S_ID=Stationary.S_ID;
(viii) Select StationaryName, Price*3 From Stationary;

5. Consider the following tables GARMENT and FABRIC. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table : GARMENT

GCODE	DESCRIPTION		PRICE	FCODE	READYDATE
10023	PENCIL	SKIRT	1150	F03	19–DEC–08
10001	FORMAL	SHIRT	1250	F01	12–JAN–08
10012	INFORMAL	SHIRT	1550	F02	06–JUN–08
10024	BABY	TOP	750	F03	07–APR–07
10090	TULIP	SKIRT	850	F02	31–MAR–07
10019	EVENING	GOWN	850	F03	06–JUN–08
10009	INFORMAL	PANT	1500	F02	20–OCT–08
10007	FORMAL	PANT	1350	F01	09–MAR–08
10020	FROCK		850	F04	09–SEP–07
10089	SLACKS		750	F03	20–OCT–08

Table : FABRIC

FCODE	TYPE
F04	POLYSTER
F02	COTTON
F03	SILK
F01	TERELENE

(i) To display GCODE and DESCRIPTION of a each dress in descending order of GCODE.
(ii) To display the details of all the GARMENTs, which have READYDATE in between 08–DEC–07 and 16–JUN–08 (inclusive of both the dates).
(iii) To display the average PRICE of all the GARMENTs, which are made up of FABRIC with FCODE as F03.

(iv) To display FABRIC wise highest and lowest price of GARMENTs from DRESS table.
(Display FCODE of each GARMENT along with highest and lowest price)
(v) SELECT SUM (PRICE) FROM GARMENT WHERE FCODE= 'F01';
(vi) SELECT DESCRIPTION, TYPE FROM GARMENT, FABRIC WHERE
GARMENT.FCODE = FABRIC. FCODE AND GARMENT. PRICE>=1260;
(vii) SELECT MAX (FCODE) FROM FABRIC;
(viii) SELECT COUNT (DISTINCT PRICE) FROM FABRIC;

6. Consider the following WORKERS and DESIG. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (vi)

WORKERS

W_ID	FIRSTNAME	LASTNAME	ADDRESS	CITY
102	Sam	Tones	33 Elm St.	Paris
105	Sarah	Ackerman	440 U.S. 110	New York
144	Manila	Sengupta	24 Friends Street	New Delhi
210	George	Smith	83 First Street	Howard
255	Mary	Jones	842 Vine Ave.	Losantiville
300	Robert	Samuel	9 Fifth Cross	Washington
335	Henry	Williams	12Moore Street	Boston
403	Ronny	Lee	121 Harrison St.	New York
451	Pat	Thompson	11 Red Road	Paris

DESIG

W_ID	SALARY	BENEFITS	DESIGNATION
102	75000	15000	Manager
105	85000	25000	Director
144	70000	15000	Manager
210	75000	12500	Manager
255	50000	12000	Clerk
300	45000	10000	Clerk
335	40000	10000	Clerk
403	32000	7500	Salesman
451	28000	7500	Salesman

(i) To display the content of workers table in ascending order of first name.
(ii) To display the FIRSTNAME, CITY and TOTAL SALARY of all Clerks from the tables workers and design, where TOTAL SALARY = SALARY + BENEFITS.
(iii) To display the minimum SALARY among Managers and Clerks from the table DESIG.
(iv) Increase the BENEFITS of all Salesmen by 10% in table DESIG.
(v) SELECT FIRSTNAME, SALARY FROM WORKERS, DESIG WHERE DESIGNATION = 'Manager' AND WORKERS.W_ID = DESIG.W_ID;
(vi) SELECT DESIGNATION, SUM(SALARY) FROM DESIG
GROUP BY DESIGNATION HAVING COUNT(*)>=2 ;

UNIT-4 BOOLEAN LOGIC

Boolean algebra is an algebra that deals with Boolean values((TRUE and FALSE) . Everyday we have to make logic decisions: “Should I carry the book or not?” , “Should I watch TV or not?” etc.

Each question will have two answers yes or no, true or false. In Boolean Algebra we use 1 for true and 0 for false which are known as truth values.

Truth table:

A truth table is composed of one column for each input variable (for example, A and B), and one final column for all of the possible results of the logical operation that the table is meant to represent (for example, A XOR B). Each row of the truth table therefore contains one possible configuration of the input variables (for instance, A = true B = false), and the result of the operation for those values.

Logical Operators:

In Algebraic function e use +,-,*,/ operator but in case of Logical Function or Compound statement we use AND,OR & NOT operator.

Example: He prefers Computer Science NOT IP.

There are three Basic Logical Operator:

1. NOT
2. OR
3. AND

- **NOT Operator**—Operates on single variable. It gives the complement value of variable.

X	$\overline{\mathbf{X}}$
0	1
1	0

- **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

- **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 logic gate is an physical device implementing a Boolean function, that is, it performs a logical operation on one or more logic inputs and produces a single logic output. Gates also called logic circuits.

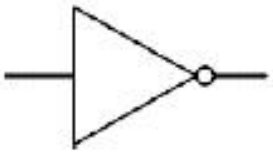
Or

A gate is simply an electronic circuit which operates on one or more signals to produce an output signal.

NOT gate (inverter): The output Q is true when the input A is NOT true, the output is the inverse of the input:

Q = NOT A

A NOT gate can only have one input. A NOT gate is also called an inverter.



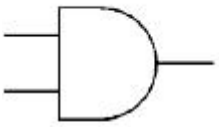
Traditional symbol

Input A	Output Q
0	1
1	0

Truth Table

AND gate

The output Q is true if input A AND input B are both true: **Q = A AND B** An AND gate can have two or more inputs, its output is true if all inputs are true.



Traditional symbol

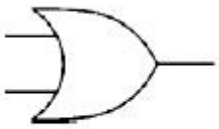
Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

OR gate

The output Q is true if input A OR input B is true (or both of them are true): **Q = A OR B**

An OR gate can have two or more inputs, its output is true if at least one input is true.



Traditional symbol

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

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 $\bar{0} = 1, \bar{1} = 0$

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) $X + \bar{X}=1$

   (b) $X. \bar{X}=0$
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

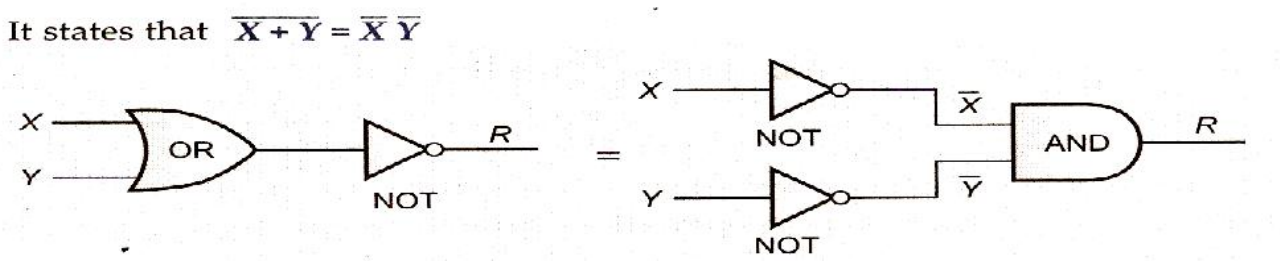
Some other rules of Boolean algebra

$X+\overline{X}Y=X+Y$

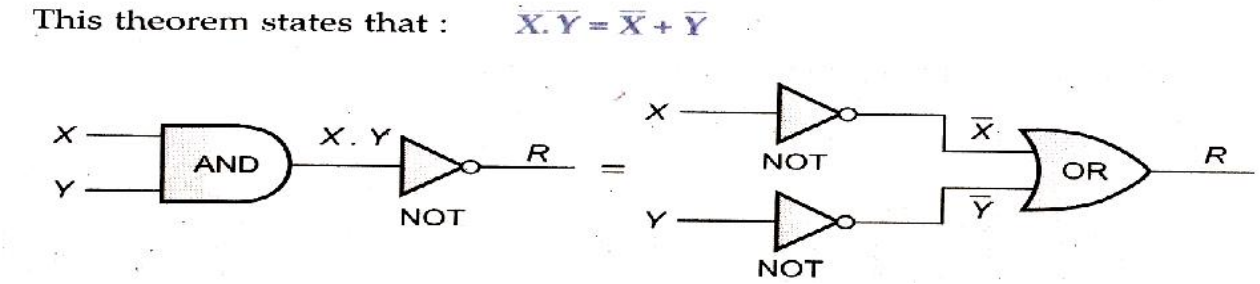
Demorgan's Theorem

A mathematician named DeMorgan developed a pair of important rules regarding group complementation in Boolean algebra.

Demorgan's First Theorem



Demorgan's Second Theorem



Derivation of Boolean expression:-

Minterm :

minterm is a Product of all the literals within the logic System.

Step involved in minterm expansion of Expression

1. First convert the given expression in sum of product form.
2. In each term is any variable is missing(e.g. in the following example Y is missing in first term and X is missing in second term), multiply that term with (missing term +complement(missing term))factor e.g. if Y is missing multiply with Y+Y")
3. Expand the expression .
4. Remove all duplicate terms and we will have minterm form of an expression.

Example: Convert X + Y

X + Y = X.1 + Y.1

=X.(Y+Y") + Y(X+X")

=XY + XY"+XY+X"Y

=XY+XY"+XY

Other procedure for expansion could be

1. Write down all the terms
2. Put X"s where letters much be inserted to convert the term to a product term
3. Use all combination of X"s in each term to generate minterms
4. Drop out duplicate terms

Shorthand Minterm notation:

Since all the letters must appear in every product, a shorthand notation has been developed that saves actually writing down the letters themselves. To form this notation, following steps are to be followed:

1. First of all, Copy original terms
2. Substitute 0s for barred letters and 1s for nonbarred letters
3. Express the decimal equivalent of binary word as a subscript of m.

Rule1. Find Binary equivalent of decimal subscript e.g.,for m6 subscript is 6, binary equivalent of 6 is 110.

Rule 2. For every 1s write the variable as it is and for 0s write variables complemented form i.e., for 110 t is XYZ. XYZ is the required minterm for m6.

maxterm:

A maxterm is a sum of all the literals (with or without the bar) within the logic system. Boolean Expression composed entirely either of Minterms or Maxterms is referred to as Canonical Expression.

Canonical Form:

Canonical expression can be represented is derived from

(i) Sum-of-Products(SOP) form

(ii) Product-of-sums(POS) form

Sum of Product (SOP)

1. Various possible input values
2. The desired output values for each of the input combinations

X	Y	R
0	0	X′Y′
0	1	X′Y
1	0	XY′
1	1	XY

Product of Sum (POS)

When a Boolean expression is represented purely as product of Maxterms, it is said to be in Canonical Product-of-Sum from of expression**.**

X	Y	Z	Maxterm
0	0	0	X+Y+Z
0	0	1	X+Y+Z′
0	1	0	X+Y′+Z
0	1	1	X+Y′+Z′
1	0	0	X′+Y+Z
1	0	1	X′+Y+Z′
1	1	0	X′+Y′+Z
1	1	1	X′+Y′+Z′

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.Algebraic method: This method makes use of Boolean postulates, rules and theorems to simplify the expression.

Example No. 1: Reduce the expression $\overline{XY} + \overline{X} + XY$.

Solution. $\overline{XY} + \overline{X} + XY$

$= (\overline{X} + \overline{Y}) + \overline{X} + XY$ *(using DeMorgan's 2nd theorem i.e.,* $\overline{XY} = \overline{X} + \overline{Y}$*)*

$= \overline{X} + \overline{X} + \overline{Y} + XY$

$= \overline{X} + \overline{Y} + XY$ $(\because\ \overline{X} + \overline{X} = \overline{X}$ as X + X = X)

$= \overline{X} + XY + \overline{Y}$

$= (\overline{X} + \overline{\overline{X}}Y) + \overline{Y} = (\overline{X} + XY) + \overline{Y}$ *(putting* $X = \overline{\overline{X}}$*)*

$= \overline{X} + Y + \overline{Y}$ $(X + \overline{X}Y = X + Y)$

$= \overline{X} + 1$ *(putting* $Y + \overline{Y} = 1$*)*

$= 1$ *(putting* $\overline{X} + 1 = 1$ as $1 + X = 1$*)*

Example No. 2: Minimise $AB + \overline{AC} + A\overline{B}C(AB + C)$.

Solution. $AB + \overline{AC} + A\overline{B}C(AB + C) = AB + \overline{AC} + A\overline{B}CAB + A\overline{B}CC$

$= AB + \overline{AC} + AAB\overline{B}C + ABCC$

$= AB + \overline{AC} + 0 + ABCC$ *(putting* $B\overline{B} = 0$*)*

$= AB + \overline{AC} + A\overline{B}.C$ *(putting* $C.C = C$*)*

$= AB + \overline{A} + \overline{C} + A\overline{B}C$ *(putting* $\overline{AC} = \overline{A} + \overline{C}$ *DeMorgan's 2nd theorem)*

$= \overline{A} + AB + \overline{C} + A\overline{B}C$ *(rearranging the terms)*

$= \overline{A} + B + C + A\overline{B}C$ *(putting* $\overline{A} + AB = A + B$ *because* $X + XY = X + Y$*)*

$= \overline{A} + \overline{C} + B + A\overline{B}C = \overline{A} + \overline{C} + B + \overline{B}AC$

$= \overline{A} + \overline{C} + B + AC$ *(putting* $B + \overline{B}AC = B + AC$ *because* $X + \overline{X}Y = X + Y$*)*

$= \overline{A} + B + \overline{C} + CA$

$= \overline{A} + B + \overline{C} + A$ $(\because\ \overline{C} + CA = \overline{C} + A)$

$= A + \overline{A} + B + \overline{C}$

$= 1 + B + \overline{C}$ *(putting* $A + \overline{A} = 1$*)*

$= 1$ *(as* $1 + X = 1$ *i.e., anything added to 1 results in 1)*

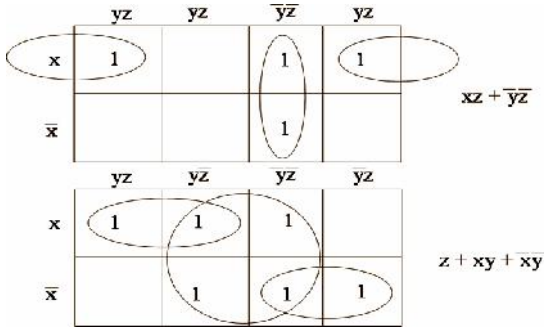
2. Using Karnaugh Map :

Karnaugh Maps:

Karnaugh map or K Map is a graphical display of the fundamental product in a truth table.

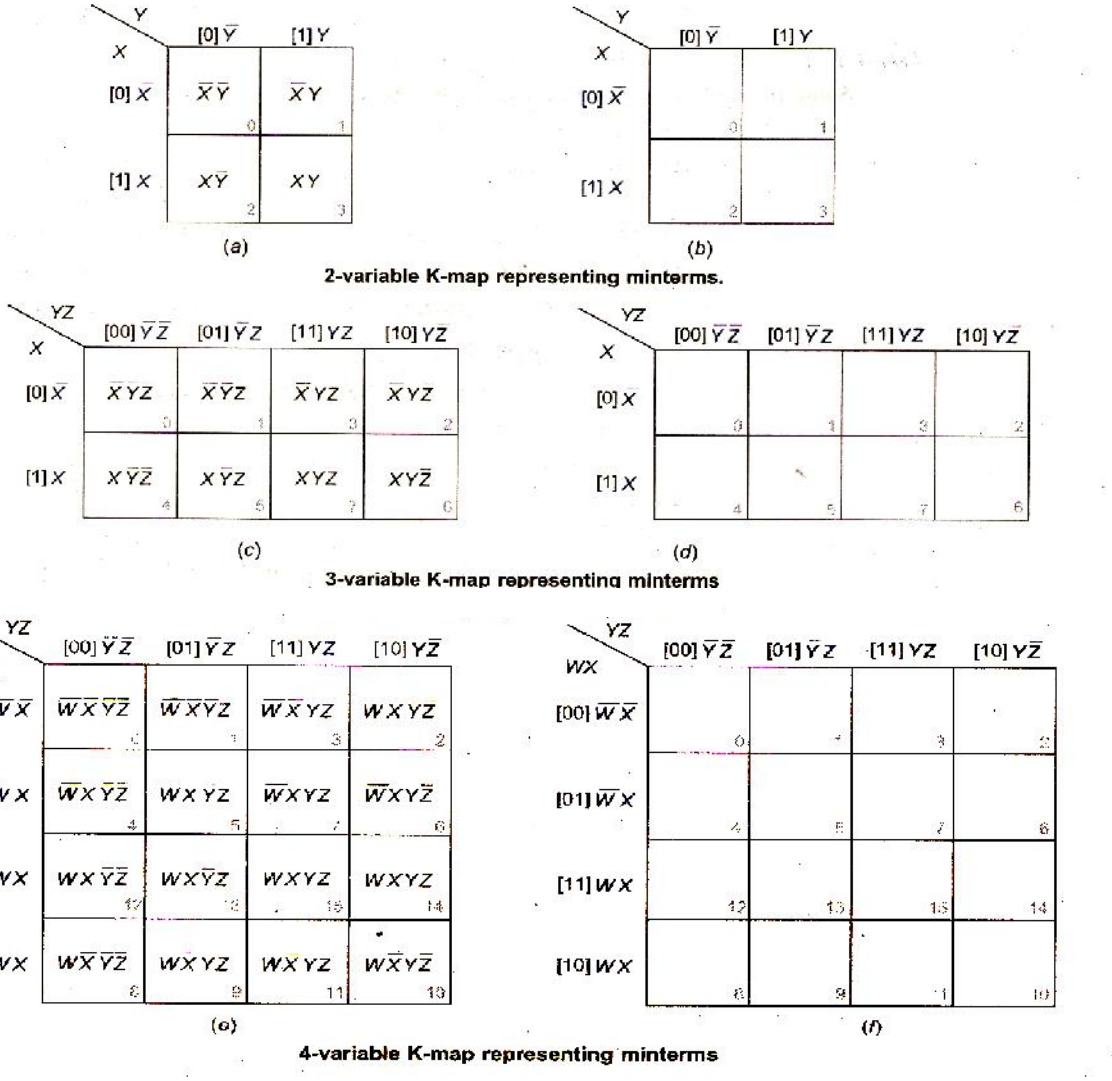
For example:

- 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 $x\,y\ +\ x\,\overline{y}\ =\ x\,(\,y\ +\ \overline{y}\,)\ =\ x$. (since y+y'=1)
- The second K-map, similarly, shows $x\,y\ +\ \overline{x}\,y\ =\ (\,x\ +\ \overline{x}\,)\,y\ =\ y$



- 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



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'.

Example1: Reduce the following Boolean expression using K-Map:
F(P,Q,R,S)=Σ(0,3,5,6,7,11,12,15)

	R'S'	R'S	RS	RS'
P'Q'	1 0	1	1 3	2
P'Q	4	1 5	1 7	1 6
PQ	1 12	13	1 15	14
PQ'	8	9	1 11	10

Soln:
This is 1 quad, 2pairs & 2 lock
Quad(m3+m7+m15+m11) reduces to RS
Pair(m5+m7) reduces to P‟‟QS
Pair (m7+m6) reduces to P‟‟QR
Block m0=P‟‟Q‟‟R‟‟S‟‟
M12=PQR‟‟S‟‟
hence the final expressions is F=RS + P‟‟QS + P‟‟QR + PQR‟‟S‟‟ + P‟‟Q‟‟R‟‟S‟‟

Example2: Reduce the following Boolean expression using K-Map:
F(A,B,C,D)=Π(0,1,3,5,6,7,10,14,15)

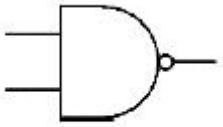
0	0	0	
	0	0	0
		0	0
			0

Soln:
Reduced expressions are as follows:
For pair 1, (A+B+C)
For pair 2, (A‟‟+C‟‟+D)
For Quad 1, (A+D‟‟)
For Quad 2, (B‟‟+C‟‟)
Hence final POS expression will be
$Y(A,B,C,D) = (A+B+C)\,(A+\overline{C}+\overline{D})\,(A+\overline{D})\,(\overline{B}+\overline{C})$

More about Gates:

NAND gate (NAND = Not AND)

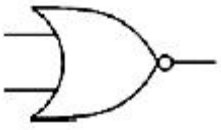
This is an AND gate with the output inverted, as shown by the 'o' on the output. The output is true if input A AND input B are NOT both true: **Q = NOT (A AND B)** A NAND gate can have two or more inputs, its output is true if NOT all inputs are true.



Input A	Input B	Output Q
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate (NOR = Not OR)

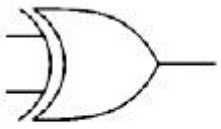
This is an OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if NOT inputs A OR B are true: **Q = NOT (A OR B)** A NOR gate can have two or more inputs, its output is true if no inputs are true.



Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

EX-OR (EXclusive-OR) gate

The output Q is true if either input A is true OR input B is true, **but not when both of them are true**: **Q = (A AND NOT B) OR (B AND NOT A)** This is like an OR gate but excluding both inputs being true. The output is true if inputs A and B are **DIFFERENT**. EX-OR gates can only have 2 inputs.

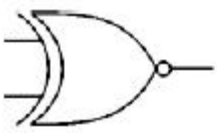


Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	0

Traditional symbol — Truth Table

EX-NOR (EXclusive-NOR) gate

This is an EX-OR gate with the output inverted, as shown by the 'o' on the output. The output Q is true if inputs A and B are the **SAME** (both true or both false):



Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	1

Traditional symbol — Truth Table

Q = (A AND B) OR (NOT A AND NOT B) EX-NOR gates can only have 2 inputs.

Summary truth tables

The summary truth tables below show the output states for all types of 2-input and 3-input gates.

Summary for all 2-input gates

Inputs		Output of each gate					
A	B	AND	NAND	OR	NOR	EX-OR	EX-NOR
0	0	0	1	0	1	0	1
0	1	0	1	1	0	1	0
1	0	0	1	1	0	1	0
1	1	1	0	1	0	0	1

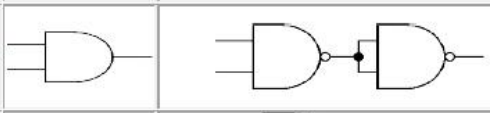
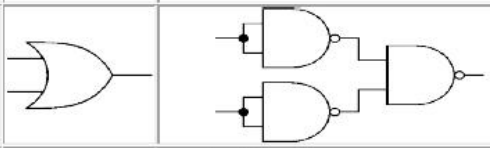
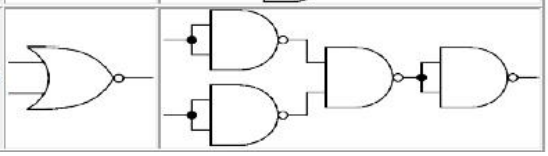
Note that EX-OR and EX-NOR gates can only have 2 inputs.

Summary for all 3-input gates

Inputs			Output of each gate			
A	B	C	AND	NAND	OR	NOR
0	0	0	0	1	0	1
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	0	1	0	1	1	0
1	1	0	0	1	1	0
1	1	1	1	0	1	0

NAND gate equivalents

The table below shows the NAND gate equivalents of NOT, AND, OR and NOR gates:

Gate	Equivalent in NAND gates
NOT	
AND	
OR	
NOR	

Low Order Thinking Questions: (Boolean Algebra)

a) State and verify absorption law in Boolean algebra.

Ans. Absorption Law states that :

a) X+XY=X b) X(X+Y)=X

b) Verify X'.Y+X.Y'=(X'+Y').(X+Y) algebraically.

Ans. LHS= X'Y + XY'

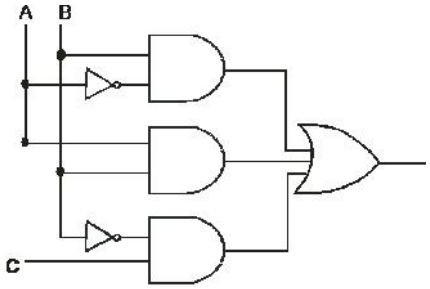
= (X'+X) (X'+Y') (Y+X) (Y+Y')

= 1.(X'+Y') (X+Y).1

= (X'+Y') (X+Y)

= RHS, hence proved

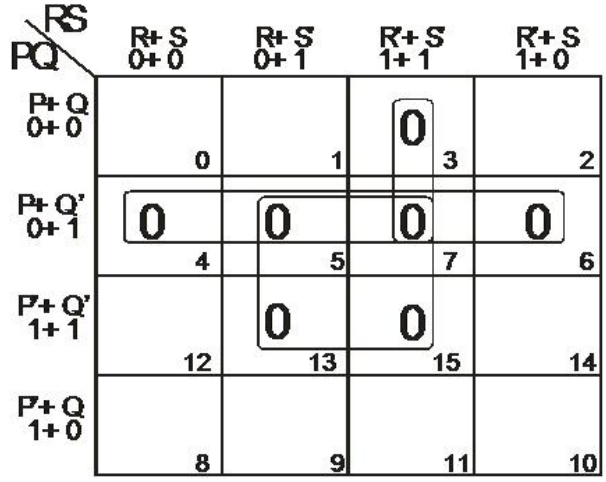
c) Write the equivalent Boolean Expression F for the following circuit diagram :



Ans.: A’B+AB+B’C

d) If F(P,Q,R,S) = ∏ (3,4,5,6,7,13,15) , obtain the simplified form using K-Map.

Ans.:



.

Reduction of groups following the reduction rule :

Quad1 = M4.M5.M6.M7

= P+Q’

Quad2 = M5.M7.M13.M15

= Q’+S’

Pair = M3.M7

= P+R’+S’

Therefore POS of F(P,Q,R,S) = (P+Q’)(Q’+S’)(P+R’+S’)

e) F(a,b,c,d)=Σ(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’

f) Write the equivalent Boolean expression for the following logic circuit:

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- Express in the product of sums form, the Boolean function F(X,Y,Z), the truth table for which is given below:

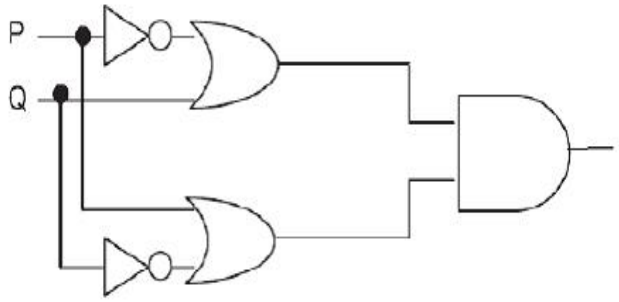
1/2 Marks Practice Questions:

1.State and Prove DeMorgan Law using Truth Table

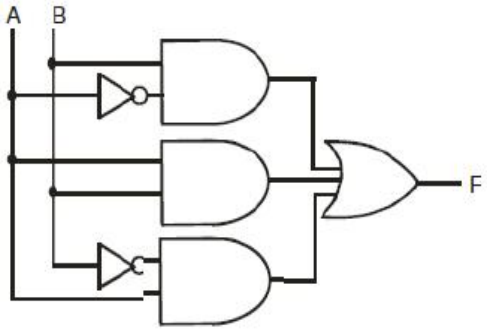
2. State and prove Absorption Law algebraically.

3. State and Prove Distributive Law algebraically.

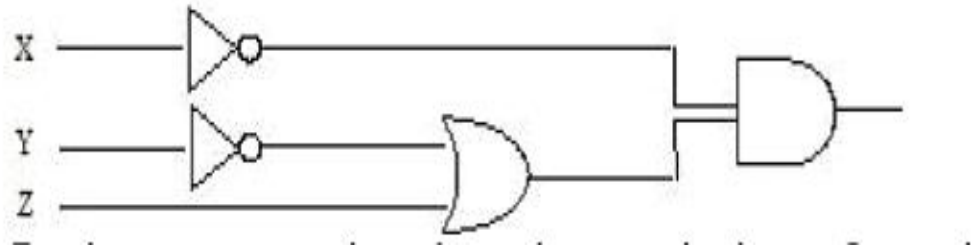
4. Write the equivalent Boolean Expression for the following Logic Circuit 2



5. Write the equivalent Boolean Expression F for the following circuit diagram : 2



6 Write the equivalent Boolean Expression F for the following circuit diagram : 2



7. 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') 1

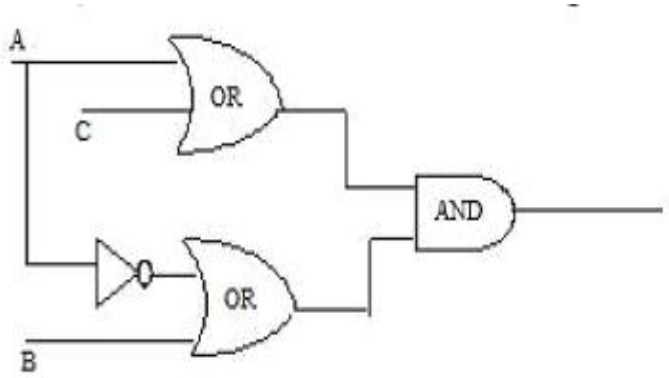
8. Convert the following Boolean expression into its equivalent Canonical Product of Sum form (POS):

A.B'.C + A'.B.C +A'.B.C' 1

9. Draw a Logical Circuit Diagram for the following Boolean expression:

A.(B+C') 2

10. Write the equivalent Boolean Expression F for the following circuit diagram: 2



11. Prove that XY+YZ+YZ'=Y algebraically 2
12. Design (A+B).(C+D) using NOR Gate. 2

3 Marks Practice Questions

13. If F(a,b,c,d)=Σ(0,2,4,5,7,8,10,12,13,15), obtain the simplified form using K-Map.

14. If F(a,b,c,d) =Σ(0,3,4,5,7,8,9,11,12,13,15), obtain the simplified form using KMap

15 Obtain a simplified form for a boolean expression

F(U,V,W,Z)= π (0,1,3,5,6,7,10,14,15)

16 . Reduce the following boolean expression using K-Map

F(A,B,C,D) = Σ(5,6,7,8,9,12,13,14,15)

UNIT 5 : COMMUNICATION AND NETWORK CONCEPTS

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.

Need

- 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

1. **ARPANET**:In 1969, The US govt. formed an agency named ARPANET (Advanced Research Projects Agency NETwork) to connect computers at various universities and defense agencies. The main objective of ARPANET was to develop a network that could continue to function efficiently even in the event of a nuclear attack.
2. **Internet (INTERconnection NETwork)**: The Internet is a worldwide network of computer networks. It is not owned by anybody.
3. **Interspace**:InterSpace is a client/server software program that allows multiple users to communicate online with real – time audio, video and text chat in dynamic 3D environments.

SWITCHING TECHNIQUES

Switching techniques are used for transmitting data across networks.
Different types are :

1. **Circuit Switching**: In the Circuit Switching technique, first, the complete end-to-end transmission path between the source and the destination computers is established and then the message is transmitted through the path. The main advantage of this technique is guaranteed delivery of the message. Mostly used for voice communication.

2. **Message Switching**: In the Message switching technique, no physical path is established between sender and receiver in advance. This technique follows the store and forward mechanism.
3. **Packet Switching**: In this switching technique fixed size of packet can be transmitted across the network.

Comparison between the Various Switching Techniques: Criteria	Circuit Switching	Message Switching	Packet Switching
Path established in advance	Yes	No	No
Store and forward technique	No	Yes	Yes
Message follows multiple routes	No	Yes	Yes

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.

Transmission media:

1. Twisted pair cable: - It consists of two identical 1 mm thick copper wires insulated and twisted together. The twisted pair cables are twisted in order to reduce crosstalk and electromagnetic induction.

Advantages:

(i) It is easy to install and maintain.

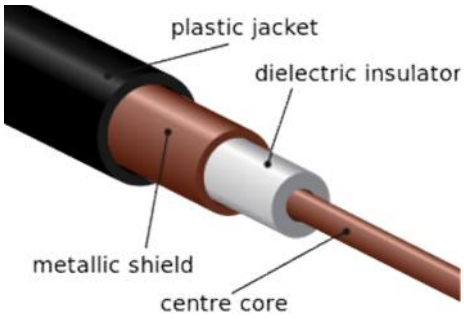
(ii) It is very inexpensive

Disadvantages:

(i) It is incapable to carry a signal over long distances without the use of repeaters.

(ii) Due to low bandwidth, these are unsuitable for broadband applications.

2. Co-axial Cables: It consists of a solid wire core surrounded by one or more foil or braided wire shields, each separated from the other by some kind of plastic insulator. It is mostly used in the cable wires.



Advantages:

(i) Data transmission rate is better than twisted pair cables.

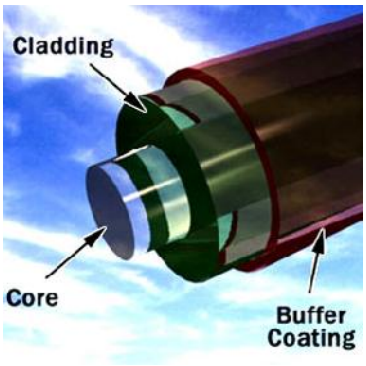
(ii) It provides a cheap means of transporting multi-channel television signals around metropolitan areas.

Disadvantages:

(i) Expensive than twisted pair cables.

(ii) Difficult to manage and reconfigure.

3. Optical fiber: - An optical fiber consists of thin glass fibers that can carry information in the form of visible light.



Advantages:

(i) **Transmit data over long distance with high security**.

(ii) **Data transmission speed is high**

(iii) Provide better noise immunity

(iv) Bandwidth is up to 10 Gbps.

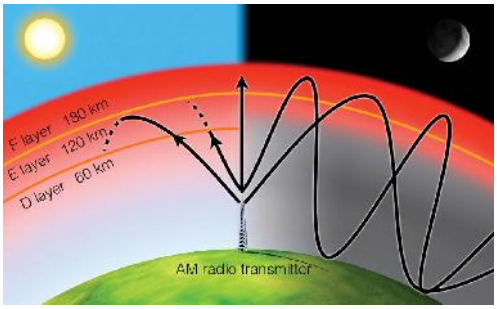
Disadvantages:

(i) Expensive as compared to other guided media.

(ii) Need special care while installation?

4. Infrared: - The infrared light transmits data through the air and can propagate throughout a room, but will not penetrate walls. It is a secure medium of signal transmission. The infrared transmission has become common in TV remotes, automotive garage doors, wireless speakers etc.

5. Radio Wave: - Radio Wave an electromagnetic wave with a wavelength between 0.5 cm and 30,000m. The transmission making use of radio frequencies is termed as radio-wave transmission



Advantages:

(i) Radio wave transmission offers mobility.

(ii) It is cheaper than laying cables and fibers.

(iii) It offers ease of communication over difficult terrain.

Disadvantages:

(i) Radio wave communication is insecure communication.

(ii) Radio wave propagation is susceptible to weather effects like rains, thunder storms etc.

6. Microwave Wave: - The Microwave transmission is a line of sight transmission. Microwave signals travel at a higher frequency than radio waves and are popularly used for transmitting data over long distances.

Advantages:

(i) It is cheaper than laying cable or fiber.

(ii) It has the ability to communicate over oceans.

Disadvantages:

(i) Microwave communication is an insecure communication.

(ii) Signals from antenna may split up and transmitted in different way to different antenna which leads to reduce to signal strength.

(iii) Microwave propagation is susceptible to weather effects like rains, thunder storms etc.

(iv) Bandwidth allocation is extremely limited in case of microwaves.

7. Satellite link: - The satellite transmission is also a kind of line of sight transmission that is used to transmit signals throughout the world.

Advantages:

(i) Area covered is quite large.

(ii) No line of sight restrictions such as natural mountains, tall building, towers etc.

(iii) Earth station which receives the signals can be fixed position or relatively mobile.

Disadvantages:-

(i) Very expensive as compared to other transmission mediums.

(ii) Installation is extremely complex.

(iii) Signals sent to the stations can be tampered by external interference.

Network devices:

Modem: A MODEM (MOdulator DEModulator) is an electronic device that enables a computer to transmit data over telephone lines. There are two types of modems, namely, internal modem and external modem.

RJ45 connector: - The RJ-45(Registered Jack) connectors are the plug-in devices used in the networking and telecommunications applications. They are used primarily for connecting LANs, particularly Ethernet.

Ethernet Card: - It is a hardware device that helps in connection of nodes within a network.

Hub: A hub is a hardware device used to connect several computers together. Hubs can be either active or passive. Hubs usually can support 8, 12 or 24 RJ45 ports.

Switch: A switch (switching hub) is a network device which is used to interconnect computers or devices on a network. It filters and forwards data packets across a network. The main difference between hub and switch is that hub replicates what it receives on one port onto all the other ports while switch keeps a record of the MAC addresses of the devices attached to it.

Gateway: A gateway is a device that connects dissimilar networks.

Repeater: A repeater is a network device that amplifies and restores signals for long distance transmission.

Network topologies and types

Topology :

- Topology refers to the way in which the workstations attached to the network are interconnected.

The BUS Topology: - The bus topology uses a common single cable to connect all the workstations. Each computer performs its task of sending messages without the help of the central server. However, only one workstation can transmit a message at a particular time in the bus topology.

Advantages:

(i) Easy to connect and install.
(ii) Involves a low cost of installation time.
(iii) Can be easily extended.

Disadvantages:-

(i) The entire network shuts down if there is a failure in the central cable.
(ii) Only a single message can travel at a particular time.
(iii) Difficult to troubleshoot an error.

The STAR Topology: - A STAR topology is based on a central node which acts as a hub. A STAR topology is common in homes networks where all the computers connect to the single central computer using it as a hub.

Advantages:

(i) Easy to troubleshoot
(ii) A single node failure does not affect the entire network.
(iii) Fault detection and removal of faulty parts is easier.
(iv) In case a workstation fails, the network is not affected.

Disadvantages:-

(i) Difficult to expand.
(ii) Longer cable is required.
(iii) The cost of the hub and the longer cables makes it expensive over others.
(iv) In case hub fails, the entire network fails.

The TREE Topology: - The tree topology combines the characteristics of the linear bus and the star topologies. It consists of groups of star – configured workstations connected to a bus backbone cable.

Advantages:

(i) Eliminates network congestion.
(ii) The network can be easily extended.
(iii) Faulty nodes can easily be isolated from the rest of the network.

Disadvantages:

(i) Uses large cable length.
(ii) Requires a large amount of hardware components and hence is expensive.
(iii) Installation and reconfiguration is very difficult.

Types of Networks:

LAN (Local Area Network): A Local Area Network (LAN) is a network that is confined to a relatively small area. It is generally limited to a geographic area such as writing lab, school or building. It is generally privately owned networks over a distance not more than 5 Km.

MAN (Metropolitan Area Network): MAN is the networks cover a group of nearby corporate offices or a city and might be either private or public.

WAN (Wide Area Network): These are the networks spread over large distances, say across countries or even continents through cabling or satellite uplinks are called WAN.

PAN (Personal Area Network): A Personal Area Network is computer network organized around an individual person. It generally covers a range of less than 10 meters. Personal Area Networks can be constructed with cables or wirelessly.

Network protocol

- 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. 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.

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.

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:

Viruses: Viruses are programs which replicate and attach to other programs in order to corrupt the executable codes. Virus enters the computer system through an external source and become destructive.

Worms: Worms are also self- replicating programs that do not create multiple copies of itself on one computer but propagate through the computer network. Worms log on to computer systems using the username and passwords and exploit the system.

Trojan horse: - Though it is a useful program, however, a cracker can use it to intrude the computer system in order to exploit the resources. Such a program can also enter into the computer through an e-mail or free programs downloaded through the Internet.

Spams: Unwanted e-mail (usually of a commercial nature sent out in bulk)

Cookies: Cookies are the text messages sent by a web server to the web browser primarily for identifying the user.

Firewall: A firewall is used to control the traffic between computer networks. It intercepts the packets between the computer networks and allows only authorized packets to pass.

Cyber Law: Cyber law refers to all the legal and regulatory aspects of Internet and the World Wide Web.

Cyber Crimes: Cyber crime involves the usage of the computer system and the computer network for criminal activity.

Hacking: Hacking is an unauthorized access to computer in order to exploit the resources.

Web Services:

WWW: The World Wide Web or W3 or simply the Web is a collection of linked documents or pages, stored on millions of computers and distributed across the Internet.

HTML (Hyper Text Markup Language):- HTML is a computer language that describes the structure and behavior of a web page. This language is used to create web pages.

XML (eXtensible Markup Language):- Extensible Markup Language (XML) is a meta language that helps to describe the markup language.

HTTP (Hyper Text Transfer Protocol):- A protocol to transfer hypertext requests and information between servers and browsers.

Domain Names: A domain name is a unique name that identifies a particular website and represents the name of the server where the web pages reside.

URL:- The Uniform Resource Locator is a means to locate resources such as web pages on the Internet. URL is also a method to address the web pages on the Internet. There are two types of URL, namely, absolute URL and relative URL.

Website: A collection of related web pages stored on a web server is known as a website.

Web browser: A software application that enables to browse, search and collect information from the Web is known as Web browser.

Web Servers: The web pages on the Internet are stored on the computers that are connected to the Internet. These computers are known as web servers.

Web Hosting: - Web Hosting or website hosting is the service to host, store and maintain the websites on the World Wide Web.

Web Scripting: - The process of creating and embedding scripts in a web page is known as Web Scripting. Types of Scripts:-

(i) **Client Side Scripts**: - Client side scripts supports interaction within a webpage. E.g. VB Script, Java Script, PHP (**P**HP‟S **H**ypertext **P**reprocessor).

(ii) **Server Side Scripts**: - Server side scripting supports execution at server – end. E.g. ASP, JSP, PHP

OPEN SOURCE TERMINOLOGIES

- **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.

Tips to solve Questions based on Networking

1. **Where Server should be placed**: Server should be placed in the building where the number of computers is **maximum**.

2. **Suggest a suitable cable layout of connection**: A suitable cable layout can be suggested in the following two ways:-
(i) On the Basis of Server: First the location of the Server is found out. Server is placed in that building where the number of computers are maximum (According to 80 – 20 rule). After finding the server position, each building distance is compared with the Server building directly or indirectly (taking other building in between). The shortest distance is counted whether it is through directly or indirectly.
(ii) On the Basis of Distance from each building: The distance between the each building is compared to all other buildings either directly or indirectly. The shortest distance is counted whether it is directly or through some other building.

3. **Where the following devices be placed**:
(i) **MODEM**:-
(ii) **HUB / SWITCH:- In all the wings**
(iii) **BRIDGE**:
(iv) **REPEATER**: It is used if the distances higher than 70 m. It regenerates data and voice signals.
(v) **ROUTER**: When one LAN will be connected to the other LAN.

4 Marks Questions (Communication and Network Concepts)

1. Knowledge Supplement Organisation has set up its new center at Mangalore 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

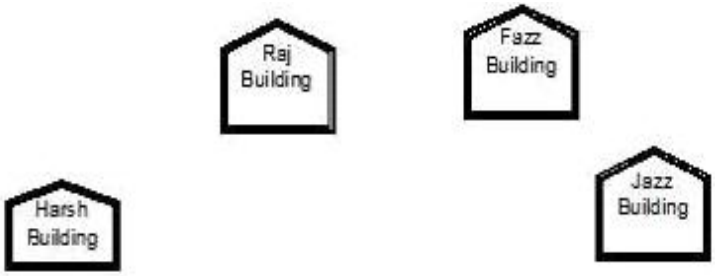
Block A to Block B	50 m
Block B to Block C	150 m
Block C to Block D	25 m
Block A to Block D	170 m
Block B to Block D	125 m
Block A to Block C	90 m

Number of Computers

Black A	25
Block B	50
Block C	125
Block D	10

- Suggest a cable layout of connections between the blocks.
- Suggest the most suitable place (i.e. block) to house the server of organisation with a suitable reason.
- Suggest the placement of the following devices with justification
    - Repeater
    - Hub/Switch
- 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?

2. Ravya Industries has set up its new center at Kaka Nagar for its office and web based activities. The company compound has 4 buildings as shown in the diagram below:



Center to center distances between various buildings is as follows:

Harsh Building to Raj Building	50 m
Raz Building to Fazz Building	60 m
Fazz Building to Jazz Building	25 m
Jazz Building to Harsh Building	170 m
Harsh Building to Fazz Building	125 m
Raj Building to Jazz Building	90 m

Number of Computers in each of the buildings is follows:

Harsh Building	15
Raj Building	150
Fazz Building	15
Jazz Bulding	25

e1) Suggest a cable layout of connections between the buildings.

e2) Suggest the most suitable place (i.e. building) to house the server of this organisation with a suitable reason.

e3) Suggest the placement of the following devices with justification:

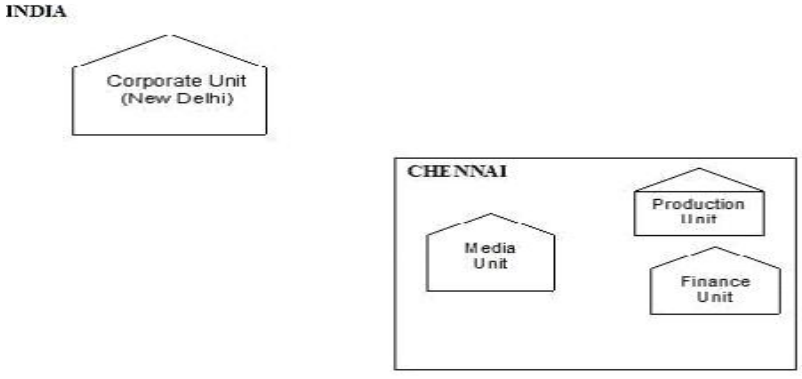
(i) Internet Connecting Device/Modem

(ii) Switch

e4) The organisation is planning to link its sale counter situated in various parts of the same city, which type of network out of LAN, MAN or WAN will be formed? Justify your answer.

3. "China Middleton Fashion" is planning to expand their network in India, starting with two cities in India to provide infrastructure for distribution of their product. The company has planned to set up their main office units in Chennai at three locations and have named their offices as "Production Unit", "Finance Unit" and "Media Unit". The company has its corporate unit in New Delhi.

A rough layout of the same is as follows:



Approximate distances between these Units is as follows:

From	To	Distance
Production Unit	Finance Unit	70 Mtr
Production Unit	Media Unit	15 KM
Production Unit	Corporate Unit	2112 KM
Finance Unit	Media Unit	15 KM

In continuation of the above, the company experts have planned to install the following number of computers in each of their office units:

Production Unit	150
Finance Unit	35
Media Unit	10
Corporate Unit	30

i) Suggest the kind of network required (out of LAN,MAN,WAN) for connecting each of the following office units:

• Production Unit and Media Unit

• Production Unit and Finance Unit

ii) Which one of the following devices will you suggest for connecting all the computers within each of their office units?

• Switch/Hub

• Modem

• Telephone

iii) Which of the following communication media, will you suggest to be procured by the company for connecting their local offices in Chennai for very effective (High Speed) communication?

• Ethernet cable

• Optical fiber

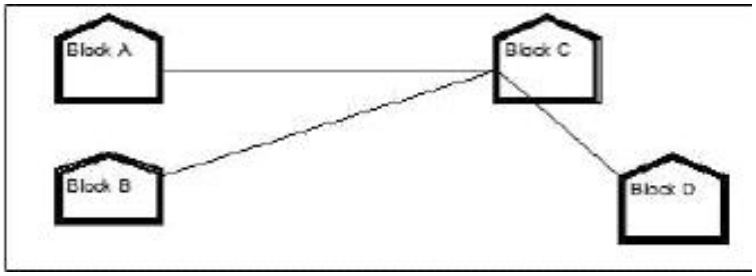
• Telephone cable

(iv) Suggest a cable/wiring layout for connecting the company's local office units located in Chennai. Also, suggest an effective method/technology for connecting the company's office unit located in Delhi.

Answers: 4 Marks (Communication and Network Concepts)

1. (e1) (Any of the following option)

Layout Option 1



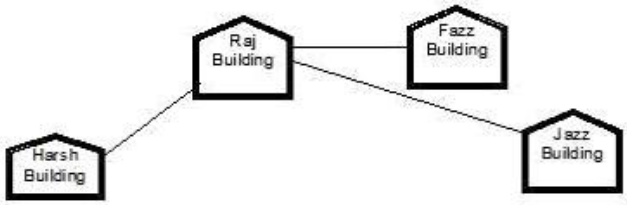
Layout Option 2

(e2) The most suitable place / block to house the server of this organisation 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.

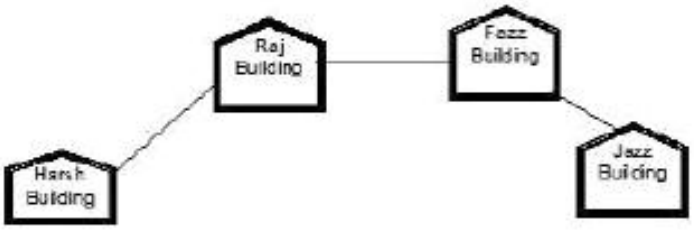
(e3) (i) For Layout 1, since the cabling distance between Blocks A and C, and that between B and C are quite large, so a repeater each, would ideally be needed along their path to avoid loss of signals during the course of data flow in these routes. For layout 2, since the distance between Blocks A and 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.

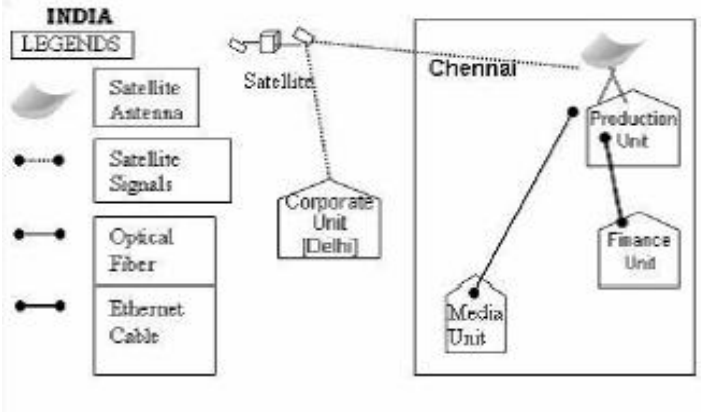
2. (e1) Any one layout



Layout option1
Layout option 2



(e2) The most suitable place / block to house the server of this organisation would be Raj Building, 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) Raj Building
(ii) In both the layouts, a hub/switch each would be needed in all the buildings, to interconnect the group of cables from the different computers in each block e4) MAN, because MAN (Metropolitan Area Networks) are the networks that link computer facilities within a city.
3. (i)(a) Production Unit and Media Unit :MAN
(b)Production Unit and Finance Unit:LAN
(ii) Switch/Hub
(iii) Optical fiber
(iv) Optical Fiber/Star Topology Wireless/Satellite Link/Leased Line



1 and 2 Marks Questions

Q(1) What do you mean by a computer network?

Ans:- Computer network is an interconnection of autonomous computers connected together using transmission media.

Q(2) What is the need for networking the computers?

Ans:- 1. Sharing of Information, 2. Reliability, 3. Reduces cost
4. Time saving

Q(3) What is the full form of ARPANET?

Ans:- Advanced Research Projects Agency Network

Q(4) What are various data transmission modes?

Ans:- There are three modes of data transmission

- Simplex
- Half-duplex
- Full-duplex

Q(5) What is the difference between Simplex and half duplex transmission?

Ans:- In simples transmission mode, the data can be transferred in only one direction where as in half duplex transmission mode, the data can be transmitted in both directions but one at a time.

Q(6) What do you mean by MODEM?

Ans:- MODEM stands for MODulatorDEModuator. It is a device that can convert an analog signal into digital signal and vice versa.

Q(7) Define the terms Bandwidth.

Ans:- Bandwidth is the range of frequencies that is available for the transmission of data. Wider the bandwidth of a communication channel, the more data it can transmit in a given period of time.

Q(8) What are various types of transmission media?

Ans:- There are two broad categories of transmission media

- Guided media
- Unguided Media

Q(9) Explain in brief the advantages and disadvantages of Twisted pair Cable.

Ans:- Advantages

- Inexpensive
- Often available in existing phone system
- Well tested and east to get

Disadvantages

- Susceptible to noise (sound, energy etc.)
- Not as durable as coaxial cable
- Does not support high speed

Q(10) What do you mean by communication protocol?

Ans:- A protocol is a set of rules to enable computers to connect with one another and to exchange information with minimum possible error.

Q(11) List various functions of Communication protocol.

Ans:- Data sequencing, Data Formatting, Flow control, Error Control,Connection Establishment and termination,Data Security

Q(12) List commonly used protocols.

Ans:- HTTP, TCT/IP, FTP, SLIP, PPP, SMTP, POP, ICMP

Q(13) What are the main functions of TCP

Ans:- The TCP does the following activities

- It breaks the data into packets that the network
- Verifies that all the packets arrived at the destination
- Reassembles the data

Q(14) What do you mean by network topology?

Ans:- Topology is how the nodes/computers are interconnected together.

Q(15) List various types of Networks.

Ans:- LAN, MAN, WAN

Q(16) Give names of various networking topologies in LAN.

Ans:- 1.Star Topology, 2.Ring topology, 3.Bus topology 4.Mesh Topology

Q(17) Write two advantages and two disadvantages of STAR topology.

Ans:- Advantages of STAR topology

- It is easy to modify and add new computers to a star network without disturbing the rest of the network.
- Troubleshooting a star topology network is easy

Disadvantages

- All the nodes are dependent on the central system. Hub. Failure of hub result in shutting down of whole of the system
- Long cable length is required

Q(18) What is NFS?

Ans:- NFS stands for Network File System. NFS is a protocol that allows a set of computers to access each others files.

HIGHER ORDER THINKING QUESTIONS

Q.1 What is protocol? How many types of protocols are there?

Ans. When computers communicate each other, there needs to be a common set of rules and instructions that each computer follows. A specific set of communication rules is called a protocol. Some protocol: PPP, HTTP, SLIP, FTP, TCP/IP

Q.2 What is the difference between Networking and Remote Networking?

Ans. The main difference between Networking and Remote Networking, is the network which we use in offices or other places locally such LAN or INTERNET and remote networking is one which we use TERMINAL Services to communicate with the remote users such WAN.

Q.3 What is point-to-point protocol?

Ans. A communication protocol used to connect computer to remote networking services include Internet Service Providers. In networking, the Point-to-Point protocol is commonly used to establish a direct connection between two nodes. Its primary use has been to connect computers using a phone line.

Q.4 How gateway is different from router?

Ans. A gateway operates at the upper levels of the OSI model and translates information between two completely different network architectures. Routers allow different networks to communicate with each other. They forward packets from one network to another based on network layer information. A gateway can interpret and translate the different protocols that are used on two distinct networks. Unlike routers that successfully connect networks with protocols that are similar, a gateway perform an application layer conversion of information from one protocol stack to another.

Q.5 What is the role of network administrator?

Ans. Basic tasks for which a network administrator may be responsible:

- Setting up and configuring network hardware and software.
- Installing and configuring network media and connections.
- Connecting user nodes and peripherals of all kinds to the network.
- Adding users to and removing users from the network.
- Managing user account.
- Ensuring the security of the network.
- Provide training to the users to utilize the network's resources.

Q.6 What is the difference between baseband and broadband transmission?

Ans. Baseband is a bi-directional transmission while broadband is a unidirectional transmission.

No Frequency division multiplexing possible in base band but possible in broadband.

SNo	Baseband	Broadband
1	Entire bandwidth of the cable is consumed by a signal	broadband transmission, signals are sent on multiple frequencies, allowing multiple signals to be sent simultaneously.
2	Digital signals	Analog signals
3	bi-directional transmission	unidirectional transmission
4	No Frequency division multiplexing possible	Frequency division multiplexing possible
5	Uses for short distance	Uses for long distance

Q.7 What are the difference between domain and workgroup?

Ans.

SNo	Domain	Workgroup
1.	One or more computers are servers	All Computers are peers.
2.	If you have a user account on the domain, you can logon to any computer on the domain.	Each computer has a set of accounts.
3.	There can be 100+ computers	Typically not more then 20-30 computers
4.	The computers can be on different local network	All computers must be on the same local netork.

Q.8 What is the differences between POP3 and IMAP Mail Server?

Ans. IMAP is a standard protocol for accessing e-mail from a local server. A simpler e-mail protocol is Post Office Protocol 3 (POP3), which download mail to the computer and does not maintain the mail on the server. IMAP, e-mails are stored on the server, while in POP3, the messages are transferred to the client's computer when they are read.

Q.9 Name different layer of the ISO OSI Model.

Ans. International Standard Orrganisation – Open Systems Interconnection has seven layers;
Physical Layer,Data Link Layer,Network Layer,Transport Layer,Session Layer,Presentation Layer
Application Layer

Q.10 What is client server architecture?

Ans. To designated a particular node which is well known and fixed address, to provide a service to the network as a whole. The node providing the service is known as the server and the nodes that use that services are called clients of that server. This type of network is called Client-Server Architecture.

Q.11 What is FDM? Give example.

Ans. FDM-Frequency Division Multiplexing is used in analog transmission. It is often used in short distance. It is code transparent and any terminal of the same speed can use the same sub-channel after the sub-channel is established. The best example if FDM is the way we receive various stations in a radio.

Q.12 Describe the following in brief:

i) MOSAIC ii) USENET iii) WAIS

Ans. i) MOSAIC: is the program for cruising the internet. The National centre wrote this program for Super Computer application at the university of Illinois. It has a simple window interface, which creates useful hypertext links that automatically perform some of the menu bar and button functions.

ii) USENET: is the way to meet people and share information. Usenet newsgroup is a special group set up by people who want to share common interests ranging from current topic to cultural heritages.

iii) WAIS: is a WIDE AREA INFORMATION SERVER.

****

SAMPLE PAPER
Class- XII
Computer Science(083)

Max Marks: 70 **Time: 3 Hrs**

General Instructions:

- *Programming Language used is C++.*
- *All questions are compulsory***.**

Q1 a)What is the difference between Call by Value and call by Reference ? Explain with code. 2

b) Write the names of header files to which the following belong : 1

setw() ii) sqrt()

c) Rewrite the following program after removing the syntactical error(s) if any 2
Underline each correction:

```
#include<iostream.h>
#include<stdio.h>
class MyStudent
{
int StudentId = 101;
char Name[20];
public:
MyStudent (){}
void Register(){cin>> StudentId;
gets name
}
void Display(){ cout << StudentId<<","<<name<<endl;
};
void main()
{MyStudent MS
Register.MS;
MS.display();
}
```

d) Find the output of the following program:

```
#include<iostream.h>
void main ( )
{int N [ ] = { 10 , 15 , 20 , 25 , 30}
int *z = N;
while (*z<30)
{
    if(*z % 3 !=0 )
    *z = *z + 2;
    else
    *z = *z + 1;
z++;
}
for (int R = 0 ; R <=4 ; R++ )
```

(3)

```
{
cout<<N[R]<<" $ ";
If (R % 3 = = 0)
cout<<endl;
}
cout<<N[4] * 3 <<endl;
}
```

e) Give the output of the following program: 2

```
#include<iostream.h>
#include<ctype.h>
void ReCode ( char Text[ ], int Num );
void main ( )
{
        char Note [ ] = "GooDLUck";
        Recode (Note, 2);
        cout << Note <<endl;
}
void ReCode (char Text [ ], int Num)
{
        for ( int  K = 0 ; Text [K] !='\0' ; K++)
                if ( K % 2 == 0)
                        Text [K] = Text [K] – Num;
                else if ( islower (Text[K] ))
                        Text [K] = toupper ( Text [K] )
                else
                        Text[K] = Text[K] + Num;
}
```

f) Study the following program and select the possible output from it : 2

```
#include <iostream.h>
#include<stlib.h>
const int MAX = 3;
void main()
{
randomize();
int Number;
Number = 50 + random (MAX);
for (int p = Number; P>=50; P--)
cout <<p<<"#";
cout <<endl;
}
```

Q2 : a) What is Function overloading ? Give an example to illustrate it. 2

b Answer the questions (i) and (ii) after going through the following class : 2

```
class Seminar
{
```

```
        int Time;
        public:
        Seminar()                    //Function 1
        {
        Time=30;cout<<"Seminar starts now"<<end1;
        }
    void Lecture() //Function 2
    {
    cout<<"Lectures in the seminar on"<<end1;
    }
    Seminar(int Duration) //Function 3
    {
    Time=Duration;cout<<"Seminar starts now"<<end1;
    }
    ~Seminar()    //Function 4
    {
    cout<<"Thanks"<<end1;
    }
    };
```

i) In Object Oriented Programming, what is Function 4 referred as and when does it get invoked/called?
ii) In Object Oriented Programming, which concept is illustrated by Function 1 and Function 3 together?
Write an example illustrating the calls for these functions.

c)Define a class **HOTEL** in C++ with the following description : private members : 4
Rno Room No of int type
Name Customer name of char type
Tarrif stores per day charges of float type
NOD no of days integer
CALC() A function to calculate and return Amount as NOD*Tarrif and if the value of NOD*Tarrif is more than 10000 then as 1.05*NOD*Tarrif
Public Members:
Checkin // A function to enter the Rno, Name , Tarrif and NOD
Checkout // A function to display Rno, Name, Tarrif, NOD and Amount by calling CALC()

d)Answer the question (i) to (iv) based on the following

```
class  Book
     {
      char Title[20];
      char Author[20];
      int noofpages;
   public:
    void read();
     void show();          };

              class TextBook: private Book
             { int noofchap, noofassignments;
             protected:
               int standard;
             public:
```

```
    void readtextbook();
    void showtextbook();
  }
  class  Physicsbook:public Textbook
  {
     char Topic[20];
      public :
      void readphysicsbook();
      void showphysicsbook();
 };
```

(i) Names the members , which are accessible from the member functions of class Physicsbook.
(ii) Write the names of members , which are accessible by an object of class Textbook.
(iii) Write the names of all members , which are accessible by an object of class Physicsbook.
(iv) How many bytes will be required by an object belonging to class Physicsbook. (4)

Q3 a) Write a function CHANGE () in C++ , which accepts an array of integer and its size as parameters and divide all those array elements by 7 which are divisible by 7 and multiply other elements by 3. (3)

b) An array T[50][20] is stored in the memory along the column with each of the elements occupying 4 bytes , find out the base address and address of elements T[30][15],if an element T[25][10] is stored at the memory location 9800. (3)

c) Write function in C++ to perform Insert operation in a dynamically allocated Queue containing names of employees . (4)

(d) Write a function in SWARARR() in C++ to swap(interchange) the first row elements with the last row elements for a two dimensional array passed as the argument of the function. (2)

(e) Evaluate the following postfix notation of expression. (2)

FALSE, TRUE, NOT, OR, TRUE, FALSE, AND , OR

Q4: a) Observe the program segment given below carefully, and answer the question that follows(1)

```
class Candidate
{
long  CId ;              //Candidate's Id
char CName[20];      // Candidate's Name
float Marks;             //Candidate's  Marks
public :
void Enter( ) ;
void Display( ) ;
void MarksChange ( );         // Function to change marks
long R_CId( )   {   return CId ; }
};
void MarksUpdate ( long  ID)
{  fstream File ;
File.open (" CANDIDATE.DAT", ios  : : binary | ios : : in  | ios : : out ) ;
Candidate  C ;
int  Record = 0 , Found = 0;
```

```
while ( ! Found && File . read ( ( char *) & C , sizeof ( C) ) )
if ( Id == C.R_CId ( ) )
{ cout << " Enter new marks" ;
C. MarkChange ( );
_____                    // Statement 1
_____                    // Statement 2
Found = 1 ;
}

Record ++ ;
}
if ( found == 1 ) cout << " Record Updated " ;
File. close ( );
}
```

Write the Statement 1 **to position** the File pointer at the beginning of the Record for which the candidate's Id matches with the argument passed, and Statement 2 **to write** the updated Record at that position.

b) Write a function in C++ to count the number of uppercase alphabets present in a text file " ARTICLE.TXT". (2)

c) Given a binary file TELEPHON.DAT, containing records of the following class Directory : (3)

```
class Directory
{
char Name [20] ;
char Address [30] ;
char AreaCode[5] ;
char 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 TELEPHON.DAT to TELEBACK.DAT.

Q 5 a) What do you mean by DDL and DML commands?Give an example of each (2)

b)Consider the following table Organisation and Grossincome and answer (I) and (II) part of the question:

Table: ORGANISATION

ECODE	NAME	POST	SGRADE	DOJ	DOB
2001	AJAY	GENERAL MANAGER	D003	23-Mar-2003	13-Jan-1980
2002	VIJAY	EXECUTIVE MANAGER	D002	12-Feb-2010	22-Jul-1987
2003	RAM	DEPUTY MANAGER	D003	24-Jan-2009	24-Feb-1983
2004	RAHIM	PROD. INCHARGE	D002	11-Aug-2006	03-Mar-1984
2005	ABBAS	ADD.GENERAL MANAGER	D001	29-Dec-2004	19-Jan-1982

Table:GROSSINCOME

SGRADE	SALARY	HRA
D001	56000	18000
D002	32000	12000
D003	24000	8000

(I) Write SQL commands for the following statements: (4)

(i) To display the details of all MEMBERS OF ORGANISATION in descending order of DOJ.

(ii) To display NAME and POST of those MEMBERS whose SGRADE is either D002 or D003.

(iii) To display the content of all the ORGANISATION table, whose DOJ is in between 09-Feb-2006 and 08-Aug-2009.

(iv) To add a new row with the following

2007, 'RUDRA', 'SALES INCHARGE', 'D002','26-Sep-2011', '26-Sept-1983'

(II) Give the output of the following SQL queries : (2)

(i) SELECT COUNT(SGRADE), SGRADE FROM ORGANISATION GROUP BY SGRADE;

(ii) SELECT MIN(DOB), MAX(DOJ) FROM ORGANISATION;

(iii) SELECT NAME,SALARY FROM ORGANISATION O, GROSSINCOME G WHERE O.SGRADE=G.SGRADE AND O.ECODE<2003;

(iv) SELECT SGRADE, SALARY+HRA FROM GROSSINCOME WHERE SGRADE='D002'

Q 6 : a) State and Verify Distributive Law. Verify using Truth Table. 2

b) Represent the Boolean expression yz+xz with the help of NAND gates only. 1

c)Obtain simplified form for a Boolean expression

$F(x,y,z,w) = \sum(1,3,4,5,7,9,11,12,13,15)$ using K Map 3

d) Write SOP form of Function F(x,y,z) whose truth table representation is given below:2

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Q7a) Compare any two Switching techniques. 1

b) Which of the following is not a Client Side script: 1

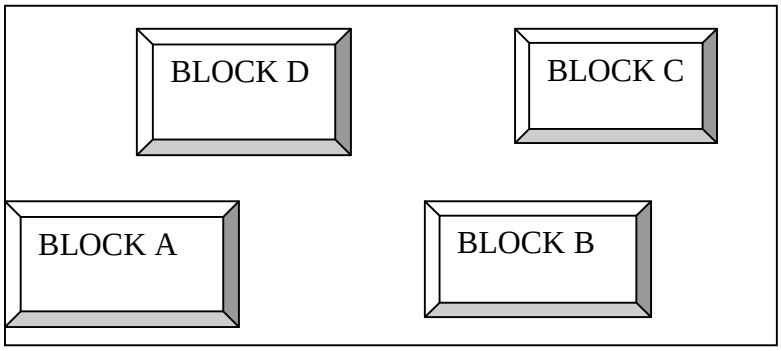
(i) VB Script (ii) Java Script

(iii) ASP (iv) PHP

c) If someone has hacked your Website, to whom you lodge the Complain? 1

d) What do you mean by IP Address? How is it useful in Computer Security? 1

e) ABC Systems Organisation has set up its new center at Jabalpur for its office and web based activities. It has 4 blocks of buildings as shown in the diagram below: 4



Center to center distances between various blocks

Block A to Block B	80 m
Block B to Block C	250 m
Block C to Block D	50 m
Block A to Block D	190 m
Block B to Block D	125 m
Block A to Block C	90 m

Number of Computers

Block A 25
Block B 50
Block C 150
Block D 10

i) Suggest a cable layout of connections between the blocks.

ii) Suggest the most suitable place (i.e. block) to house the server of this organization with a suitable reason.

iii) Suggest the placement of the following devices with justification

(i) Repeater (ii) Hub/Switch

iv) The organization is planning to link its International Office situated in Mumbai , which wired communication link , you will suggest for a very high speed connectivity?

f) What is meant by Trojan Horse and Virus in terms of computers? 1

g) What was the role of ARPANET in the communication network ? 1