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 Object 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 valid 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) It can contain Alphabets, digits or underscore
  - (ii) It must be started with alphabet or underscore (_).
  - (iii) It must not contain special characters.
  - (iv) It cannot be a Reserve word.
- **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', single character enclosed in single quotes.
  - iv. String Literal i.e cout<< "Welcome" , anything enclosed in double quotes
- Operator – performs some action on data
  - ❖ Arithmetic(+,-,*,/,%)
  - ❖ Assignment operator (=)
  - ❖ Increment / Decrement (++, --)
  - ❖ Relational/comparison (<,>,<=,>=,==,!=).
  - ❖ Logical(AND(&&),OR(||),NOT(!).
  - ❖ Conditional (? :)

Precedence of operators:

++(post increment),--(post decrement)	High
++(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. **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- Named storage location 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 be changed later on during program execution. e.g. **const int pi =3.14;**

cout – It is an object of **ostream class** defined in **iostream.h** header file and used to display value on monitor.
cin – It is an object of **istream** 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 ifelse-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
				while(expression)
				{
				statements;
				}
				Entry control loop works for true condition.
				
				do
				{
				statements;
				} while(expression);
				Exit Control Loop
				
				execute at least once if

			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
				the condition is false at beginning.
				
				for loop
				for(expression1;expressio n2;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 File	**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 =10;
  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];

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

- **String (Array of characters)** –Defined in c++ as one dimensional array of characters as char s[80]= “KV Kumbhirgram”;

S	K	V		K	u	m	b	h	i	r	g	r	a	m	\0					

Two dimensional array

A two dimensional 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 & 4columns.

	0	1	2	3
0	A[0][0]	A[0][1]	A[0][2]	A[0][3]
1	A[1][0]	A[1][1]	A[1][2]	A[1][3]
2	A[2][0]	A[2][1]	A[2][2]	A[2][3]

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 add; // nested structure
}
```

Declaration: employee e;
Input /Output : **cin>>e.add.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

(1) **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++)        //String.h
        puts(Msg);                             // stdio.h
}
```

Ans : stdio.h, string.h

(2) **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; //string.h
}
```

Ans: iostream.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
}
```

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++;
        }
  }
```

*** consider all required header file are include**

Note (index)	L	str	Output
		vR2GooD	
0	**6**	vR2Goo**d**	vR2Goo**d**
1	**5**	vR2Go**O**d	R2Go**O**d
2	**4**	vR2G**OO**d	2G**O**Od
3	**3**	vR2**g**OOd	**g**OOd

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?

```
#include<iostream.h> CBSE 2012
#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()
{
```

M

5	0	0
5	**1**	**20**

```
    METRO M(5),T;
    M.Trip();
```

T

1	0	0
1	**1**	**0**

```
     M.StatusShow();
    T.StatusShow();
    M.StatusShow();
}
```

Ans : 5: 1: 20
1: 1: 0
5: 1: 20

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 iscalled as modularity.

• It reduces the complexity to some degree and

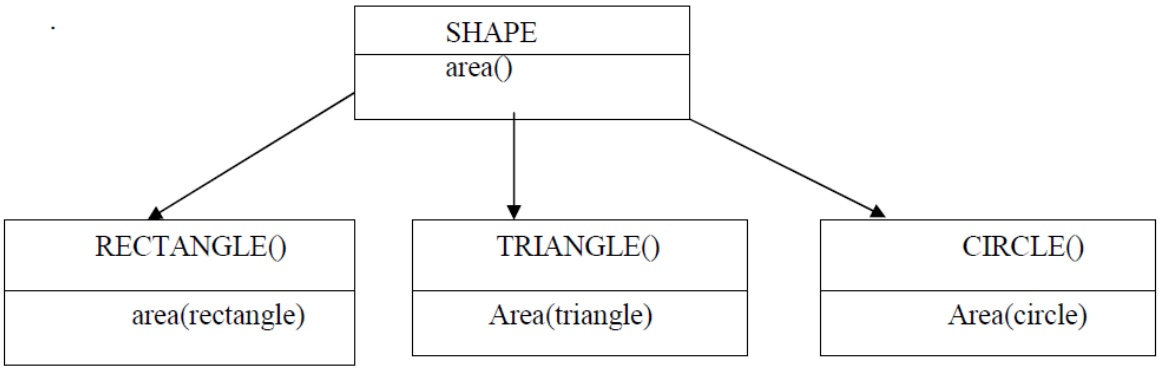
• It creates a number of well-defined boundaries within the program .

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 *state* and *behavior (data members and member 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;
```

[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
} container;                    // container is an object of class 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);

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.

```
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 Aand 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++

Structure	Class
• **Defined using struct keyword** • **Members are public by default** • **It cannot be inherited**	• **Defined using class keyword** • **Members are private by default** • **It can be inherited**

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)
{
```

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

```
        cout<<a*a;
}
void main()
{.
        Square(4); { cout <<4*4;}           // { cout <<4*4;}
        Square(8) ; { cout <<8*8; }         // { cout <<8*8; }
}
```

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 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 whenan 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.

EXAMPLE OF SINGLE INHERITANCE

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

```
// 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:

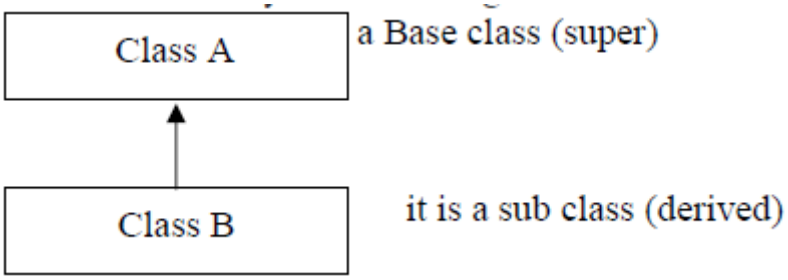
- Constructors, destructors and copy constructors of the base class.
- Overloaded operators of the base class.
- The friend functions of the base class.

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:

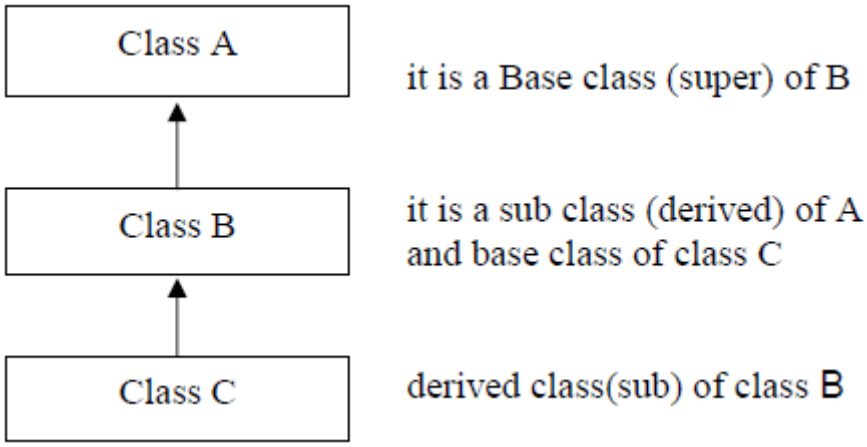
1. **Public Inheritance:** When deriving a class from a **public** base class:
   - **public** members of the base class become **public** members of the derived class and
   - **protected** members of the base class become **protected** members of the derived class.
   - A base class's **private** members 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

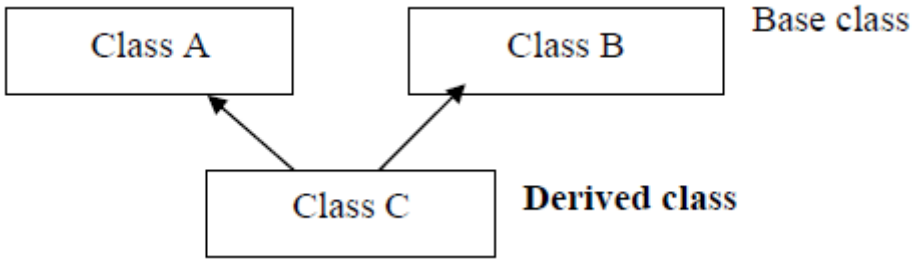
1. **Single class Inheritance:**
   - Single inheritance is the one where you have a single base class and a single derived class.



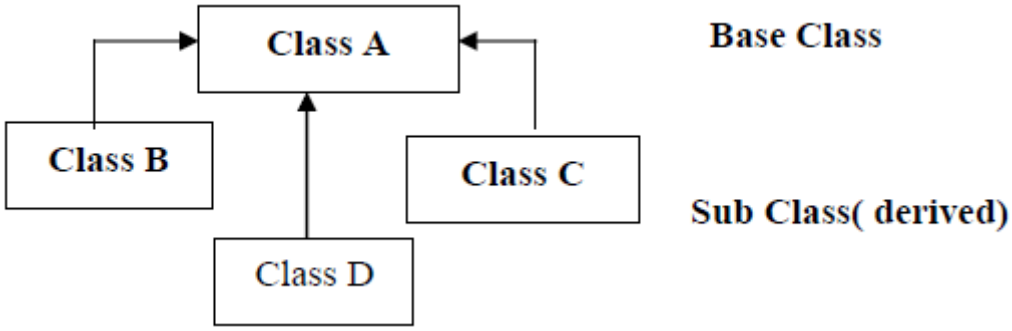
2. **Multilevel Inheritance:**
   - In Multi level inheritance, a subclass inherits from a class that itself inherits from another class.



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

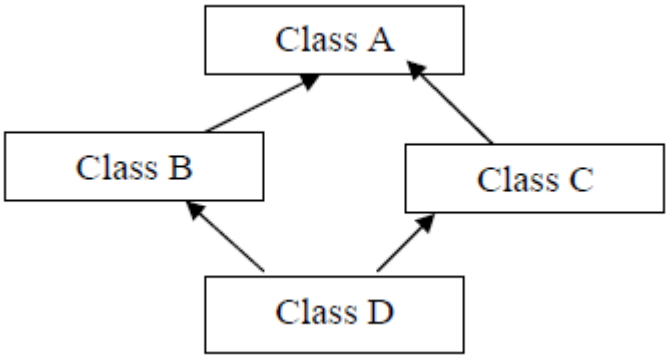


4. **Hierarchical Inheritance:**
   - In hierarchial Inheritance, it's like an inverted tree. So multiple classes inherit from a single base class.



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  // Base class WORLD & Sub class Contry
{
        int T;
protected :
        int U;
public :
        void INDATA( int, int)
        void OUTDATA();
};
class STATE : public COUNTRY    // Base Class COUNTRY & Sub Class STATE
{
        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) etchdata() 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:

Open Mode	Description
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 opens the file in write mode, and place file pointer at the end of file
ios::ate	It open the file in write or read mode, and place file pointer at the end of 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.

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, **Initial address** and **length of the variable**

File Pointer: The file pointer indicates the position in the file at which the next input/output is to occur. There **read pointer** and **write pointer** associated with a file.

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:

1. **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
}
```

2. **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;
}
```

3. **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);
}
```

4. **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;
}
```

5. **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

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

```
#include<fstream.h>
struct student
{
    char name[15];
    float percent;
};
void main()
{
    ofstream fout;  // f out is output file stream it will open file in write mode
    char ch;
    fout.open("student.dat", ios::out | ios:: binary);      // student.dat will be opened in
                                                            // binary Mode
    clrscr();
     student s;      //s is a variable of type student that contains name & percent
    do
    {       // inputting data to record  s
            cout<<"\n enter name of student";
            gets(s);
            cout<<"\n enter persentage";
            cin>>percent;
            //Writing contents of s to file student.dat
            fout.write ( ( char * ) &s, sizeof ( s ) ) ;
            cout<<"\n more record y/n";
            cin>>ch;
    }while(ch!='n' || ch!='N') ;
    fout.close();
}
```

2. **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;    //fin is an input file stream that can open a file in read mode
        student s;       // s is a record of type student
        fin.open("student.dat",ios::in | ios:: binary); // opening student.dat in binary mode
        fin.read((char *) &s, sizeof(student)); //read a record from file 'student.dat'
        invariable s
        while(file) // file will read until end of file does not come.
        {
                //Displaying the content of  s (reord) read from the student.dat
                cout<<s.name;
                cout<<"\n has the percent: "<<s.percent;
                fin.read((char *) &s, sizeof(student)); // reading the next record
        }
        fin.close();
}
```

Binary file using Objects and other file operations:

1. **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( )    // function to input values for data member of current object
        {
                cout<<"\n enter roll no";
                cin>>rno;
                cout<<"\n enter name ";
                gets(name);
                cout<<"\n enter age";
                cin>>age;
        }
        void output( )          // function to display contents of current object
        {
                cout<< "\n roll no:"<<rno;
                cout<< "\n name :"<<name;
                cout<< "\n age:"<<age;
        }
        int getrno( )           // function to get  the rno from current object
```

```
        {
                return rno;
        }
};

void create_file( )       // function to create a blank file student.dat
{
        ofstream fout;  // fout is output file stream object that will open a file in write
mode
        fout.open("student", ios::out | ios:: binary); // opening file in binary mode
        fout.close();     // closing file student.dat
}

void read_file( )          //function to read records from student.dat one by into record s
{
        ifstream fin;
        student s;
        fin.open("student.dat",ios::in | ios:: binary); //opening the file
        fin.read((char *) &s,sizeof(student));      //reading first record from file to record s
        while(file)
        {
                s.output();         // displaying the content of object s (record)
                cout<< "\n";
                fin.read ( ( char * ) & s,sizeof ( student ) ) ; // reading next record
        }
        fin.close();
}

void modify_record()
{
        student s;
        fstream file;
        file.open("student.dat",ios::in|ios::out|ios::ate|ios::binary); // opening student.dat in read
        &
                                                                    //write binary mode
        int r, pos = -1, f=0;
        cout<<"\n enter the rollo no of student whom data to be modified";
        cin>>r ;
        //Searching the record with rno = r
        file.read( ( char * )&s, sizeof( s ) ) ;
        while(file)
        {
                if (r == s.getrno( )) // will be true if required record found (rno = r)
                {
                        f=1;      // f = 1 indicate record found
                        cout<<"\n record is ";
                        s.output();
                        pos =file.tellg()-size(s); //moving the write pointer one record back
                        break;
```

```
                }
                file.read((char *)&s,sizeof(s));   // writing the modified record to the file
        }
        if(f == 0 )         // f==0 indicate record did not found
                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 = 0;
        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==0)
                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;
47
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

1. **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.**

   class Phone **CBSE 2012**

   {

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

Ans 1 :

```
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
}};
```

2. **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.2.

```
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();
}
}
```

3. **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 3.

```
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.

C++MemoryMap :

- 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.

StaticMemory Allocation : Allocation of memory at compile time.

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

Dynamic Memory Allocation : allocation of memory at run time using operator **new and delete**.

```
e.g int x =new int;      // dynamic allocation
float y= new float;
delete x;                //dynamic deallocation
delete y;
```

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:

Syntax : Datatype *variable_name;

```
int *p;      // p is an integer pointer contains address of an integer variable
float *p1;   // p1 is a float pointer contains address of a floating point variable
char *c;     // c is a character pointer contains address of a character variable

int a = 10;          // a is an integer variable hold value 10
int *p = &a;         //pointer initialization [ p is a integer pointer  holds address of a]
                     // &a means address of a
```

Pointer arithmetic:

Two arithmetic operations, addition and subtraction, may be performed on pointers.
Adding 1 to a pointer actually adds the size of pointer's base type.
Base address: The address of the first byte is known as BASE ADDRESS.

Dynamic Allocation Operators:

int * p = new int[10]; this will dynamically allocate 20 bytes (10*2) to integer pointer p

Two dimensional arrays:

```
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:

```
delete arr;
```

Pointers and Arrays:

Array of Pointers:

To declare an array holding 5 int pointers –

```
int * ip[5];
```

That would be allocated for 5 pointers that can point to integers.

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

I p	Address of a	Address of b	Address of c	Address of d	Address of e

Pointers and Strings:

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

```
    void main()
    {
            char str[ ] = "computer";
            char *cp;
            cp = str;           // cp will hold address of first element of array str
            cout<<str ;         //display string
            cout<<cp;           // display string
            for (cp =str; *cp != '\0'; cp++) // display character by character by character
            cout << "--"<<*cp;
    }
```

Output :

Computer

Computer

--c--o--m--p--u--t--e—r

Pointers and CONST :

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

```
Int n = 20;
```

int * const c = &n; // a constant pointer c to an integer n

A **pointer to a constant** refers to a pointer which is pointing to a symbolic constant.

```
const int b = 10;          // a constant integer b
const int *pc = &b;    // a pointer to a constant  integer  b
```

Pointers and Functions :

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.

```
#include<iostream.h>
void swap(int *m, int *n)
{
        int temp;
        temp = *m;
        *m = *n;
        *n = temp;
}

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;
}
```

Input :
Value of a : 5 and b : 6
After swapping value of a : 6 and b : 5

Function returning Pointers :

A function can also returns a pointer.

Syntax:- type * function-name (argument list);

```
#include <iostream.h>
int *min(int  &x,  int  &y)      // function returning int pointer
{
        if (x < y )
                return (&x);
        else
                return (&y)
}
void main()
{
        int a, b, *c;
        cout << "\nEnter a :"; cin >> a;
        cout << "\nEnter b :"; cint >> b;
```

```
    c = min(a, b);  // pointer returned from the function min will be assigned to pointer variable c
    cout << "\n The minimum no is :" << *c;
}
```

Dynamic structures:

- **Dynamic allocation:-**

  The new operator can be used to create dynamic structures also.

```
struct  student
{
        int rno;
        char name[20];
};
student *s = new student;
```

  To access structure member through structure pointer we use **arrow operator**(->).

```
cout << s -> rno;
cout << s -> name
```

- **Dynamic deallocation:-**

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

```
delete stu;
```

this Pointer : refers to the address of current object.

Solved Questions

Q1. 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.

Q2. 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.

Q3. 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 :

- area to hold the compiled program code
- area to hold global variables
- 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.
- 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.

Q7. 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.

(ii) 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++;
}r
eturn 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; }
```