Assignment No.1

1 Enlist memory management operators.

There are two types of memory management operators in C++:

- New - The new operator in C++ is used for dynamic storage allocation. This operator can be used to create object of any type
- Delete - The delete operator in C++ is used for releasing memory space when the object is no longer needed.

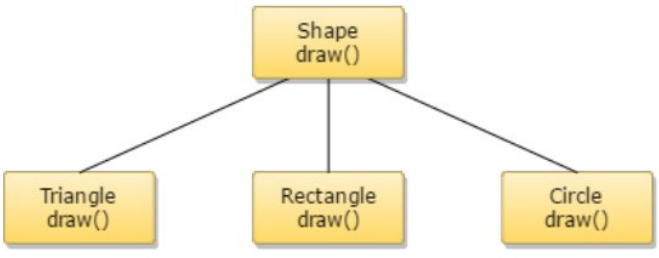
These two memory management operators are used for allocating and freeing memory blocks in efficient and convenient ways.

2 Enlist memory dereferencing operators.

A dereference operatoroperates on a pointer variable, and returns the location-value, or l-value that it points to in memory. ' * ' is the dereference operator and can be read as "value pointed by".

3 Describe any four concepts of object oriented programming.

- Objects - Objects are basic run time entities in an object oriented system. They may also represent user defined data such as vectors, time and lists. Program object should be chosen such that they match closely with the real world object. The object has state, behaviour, responsibility, identity which represents the object.
- Classes - Class is like blueprint of an object. This doesn't actually define the data but it does define what the class name means i.e. what the object of the class will consist of and what operations can be performed on such object. It is also called as user defined data type (UDT). E.g. Fruit Mango , here Fruit a class and Mango is an object of type fruit.
- Inheritance - Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. Inheritance provides re usability. This means that we can add additional features to an existing class without modifying it.
- Polymorphism – Polymorphism, a geek term means the ability to take more than one form. An operation may exhibit different behaviour is different instances. The behaviour depends upon the types of data used in operation.



Polymorphism

- Dynamic Binding – Binding refers to the linking of procedure call to the code to be executed in response to the call. 'Dynamic Binding' means that the code associated with a given procedure call is not known until time of call at run time.
- Message Passing – An object oriented programming consists of set of objects that communicate with each other. The process of programming in an object oriented language, involves the basic steps:
  i. Creating classes that define their object and their behaviour.
  ii. Creating object from class definition.

iii. Establishing communication among objects.

- Data Abstraction and Encapsulation –'Abstraction' refers to the act of representing essential features without including background details or explanation. The wrapping up of data and function into single unit is known as 'encapsulation'. Data is not accessible to the outside world and only those function wrapped in the class can access it called 'data hiding' or 'information binding'.

4 Describe any four benefits of object oriented programming.

- Simplicity – Objects is an object oriented program model real world entities, so it is easier to solve real world problems.
- Modularity – Each object forms a separate entity whose internal structure is hidden from other parts of the system.
- Modifiability – It is easy to make changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the external world can access the class through the use of functions which is its public interface.
- Extensibility – Adding new features or responding to changing requirements can be solved by introducing new classes using existing ones. Object-oriented systems can be easily upgraded from small to large systems.
- Re-usability – Objects can be reused in different programs.
- Maintainability – Objects are maintained separately which makes it easy to locate and correct problem.
- Flexibility – Polymorphism enables an object to behave differently according to the context.
- Security – Due to encapsulation, data is insulated from external access. This avoid misuse of data.

5 Describe any four features of object oriented programming.

- OOP ties data closely to functions that operate on it and protects it from accidental modification from outside functions by giving security to data.
- OOP treats data as critical element in programming and does not allow it to flow freely around the system.
- In OOP, problem is divided into number of objects and builds data and function around these objects.
- Data of an object can be accessed only by function associated with object.
- Emphasis is on data rather than procedure.
- Objects may communicate with each other through functions.
- New data and functions can be easily added whenever necessary.
- OOP follows bottom-up approach in program designing.

6 Enlist any eight or four application of object of object oriented programming.

Applications of OOP:

- Real time systems
- Simulation and modeling
- Artificial intelligence and expert system
- Object oriented database
- Neural network and parallel programming
- Decision support and office automation system
- In CAD/CAM system
- Compiler construction
- Developing Graphical User Interface
- System Application
- Operating system

7 Differentiate between OOP and POP (any eight points)

OOP	POP
OOP follows Bottom Up approach.	POP follows Top Down approach
In OOP, program is divided into parts called objects	In POP, program is divided into small parts called functions.
In OOP, Importance is given to the data rather than procedures or functions because it works as a real world.	In POP, Importance is not given to data but to functions as well as sequence of actions to be done.
In OOP, objects can move and communicate with each other through member functions.	In POP, Data can move freely from function to function in the system.
In OOP, data cannot move easily from function to function, it can be kept public or private so we can control the access of data.	In POP, Most function uses Global data for sharing that can be accessed freely from function to function in the system
OOP provides Data Hiding so provides more security	POP does not have any proper way for hiding data so it is less secure.
In OOP, overloading is possible in the form of Function Overloading and Operator Overloading	In POP, Overloading is not possible.
Examples of OOP are: C++, JAVA, VB.NET, C#.NET.	Example of POP are : C, VB, FORTRAN, Pascal

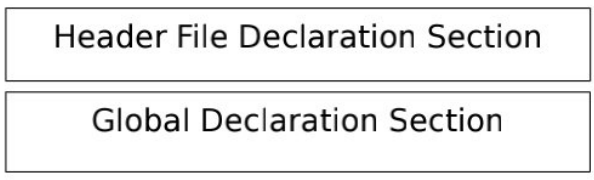
8 Write a program to show use of scope resolution operator without using class and object.

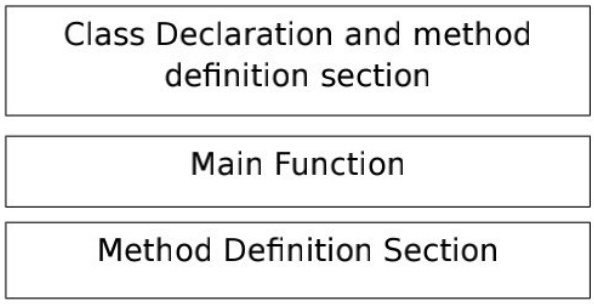
```
#include<iostream>
usingnamespacestd;

int x = 20;  // Global x

int main()
{
int x = 10; // Local x
cout<<"Value of global x is "<< ::x;
cout<<"\nValue of local x is "<< x<<endl;
  ::x++;
x++;
cout<<"Value of global x after incriment "<< ::x;
cout<<"\nValue of local x is after incrimentv"<< x<<endl;
return 0;
}
```

9 Describe structure of c++program.





Section 1: Header File Declaration

- Header files used in the program are listed in this section.
- A header file provides prototype declaration for different library functions.
- We can also include user defined header file.

Section 2: Global Declaration Section

- Global Variables are declared in this section.
- Global declaration may include:
    1. Declaring structure
    2. Declaring class
    3. Declaring Variables

Section 3: Class Declaration Section:

- Actually, this section can be considered as a sub section for global declaration section.
- Class declaration and all methods of that class are defined in this section.

Section 4: Main Function

- Each and every c++ program always starts with main function.
- This entry point for all function. Each and every method is called indirectly through main.
- We can create class and object in main.
- Operating system calls this function automatically.

Section 5: Method Definition Section

- This is optional section. In this section generally user defined function are available.

Assignment No. 2

1. Define

   i) Class -Class is like blueprint of an object. This doesn't actually define the data but it does define what the class name means i.e. what the object of the class will consist of and what operations can be performed on such object. It is also called as user defined data type (UDT).

   ii) Objects - Objects are basic run time entities in an object oriented system. They may also represent user defined data such as vectors, time and lists. Program object should be chosen such that they match closely with the real world object. The object has state, behaviour, responsibility, identity which represents the object.

2. Define or explain

   i) Static data members – Members which are associated with the class itself rather than individual objects are called as static members. A data member of a class can be made static. A static variable is initialized to zero when object are created.
   Syntax –
   A. For defining static member variable inside class:
      staticdata_type variable name;
      Example- Static int count;
   B. For defining static member variable outside class:
      data_type class-name ::static variable;
      Example- int Test :: count;

   ii) Static member function – A static member function can have access to only other static variable or functions declared in the same class. It can be called using the class name instead of object. It can be declared inside the class with static keyword placed before return type. A static member function does not have this pointer. They may not be virtual. There can't be static and non static version of same function.
   Syntax –

```
Static return_typefunction_name()
{
}
```

   Syntax for calling static function:

```
class_name :: function_name();
```

3. Define friend function

   A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions. A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.

4. Define Inline function

   Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the C++ compiler at compile time. Inline function may increase efficiency if it is small.

   The syntax for defining the function inline is:

```
inlinereturn_typefunction_name(parameters)
```

```
{
        //function code
}
```

5. State the syntax to define the function outside of the class.

A public member function can also be defined outside of the class with a special type of operator known as Scope Resolution Operator (SRO); SRO represents by :: (double colon)
Syntax –

```
return_typeclass_name::function_name(parameters)
{
        function body;
}
```

6. State the syntax to define inline function .

```
inline return-type function-name(parameters)
{
  // function code
}
```

7. State the merits and demerits of defining inline function.

Merits:

1) It does not require function calling overhead.
2) It also save overhead of variables push/pop on the stack, while function calling.
3) It also save overhead of return call from a function.
4) It increases locality of reference by utilizing instruction cache.

Demerits:

1) May increase function size so that it may not fit on the cache, causing lots of cahce miss.
2) After in-lining function if variables number which are going to use register increases than they may create overhead on register variable resource utilization.
3) It may cause compilation overhead as if some body changes code inside inline function than all calling location will also be compiled.
4) If used in header file, it will make your header file size large and may also make it unreadable.

8. Enlist various access specifiers.

- public
- private
- protected

9. State by default what is the status of data members in class? Give example.

By default, the status of data members in class is private.
Example-

```
 classabc
{
        int X;
 public:
        voiddispX();
};
```

Here, data member ‘X’ is a private member of class abc. Any access specifier is not used before ‘X’ , therefore it is private by default.

10. Develop a program to create a class college which has name, code and address as data member and getdata() and showdata() as member function to read and display information.

```
#include<iostream>
usingnamespacestd;

class college
{
        char name[25], address[50];
        int code;
public:
        voidgetdata()
        {
                cout<<"Enter name, address and code: ";
                cin.getline(name,25);
                cin.getline(address,50);
                cin>>code;
        }

        voidshowdata()
        {
                cout<<"\nName: "<<name;
                cout<<"\nAddress: "<<address;
                cout<<"\nCode: "<<code<<endl;
        }
};
int main()
{
        college C;
        C.getdata();
        C.showdata();
        return 0;
}
```

11. Explain array of objects OR Develop a program to create a class employee which has name, id and salary as data member and getdata() to read information for 10 employees and showdata() as member function to display name and id of those employees whose salary is more than 10000.

Array of objects: An object of class represents a single record in memory, if we want more than one record of class type, we have to create an array of class or object. Array is a collection of similar type of data's that are referenced by a common name. Similarly, array of class objects means a collection of similar type of objects referenced by a common name. Hence, an array having class type element is known as array of object.

Syntax for Array of object:

Class_nameobject_name [size];

Where, size is a size of the array of class objects.

```
#include<iostream>
usingnamespacestd;

class employee
{
        char name[20];
        int id, salary;
public:
        voidgetdata();
        voidshowdata();
};
```

```
void employee::getdata()
{
        cout<<"\nEnter name, id and salary of employee:\n";
        gets(name);
        cin>>id>>salary;
}
void employee:: showdata()
{
        if(salary>10000)
                cout<<"\nName: "<<name<<"\nID: "<<id;
}
int main()
{
        employee e[10];
        int i;
        cout<<"\nEnter details of 10 employees\n";
        for(i=0; i<10; i++)
        {
                e[i].getdata();
        }
        cout<<"\nEmployees having salary more than 10000";
        for(i=0; i<10; i++)
        {
                e[i].showdata();
        }
        return 0;
}
```

12. Write a program to create a class distance which has data members as feet and inch and member function ad getd() and showd() to read and display result. Implement sum() as friend function to perform addition between two objects . (Perform conversions from inches to feet)

```
#include<iostream>
usingnamespacestd;

classdist
{
        intfeet,inch;
public:
        voidgetd();
        voidshowd();
        frienddist sum(dist, dist);
};
voiddist :: getd()
{
        cout<<"Enter distance (in feet and inches):\n";
        cin>>feet>>inch;
}
dist sum(dist A, dist B)
{
        dist d;
        d.inch = A.inch + B.inch;
        d.feet = d.inch / 12;
        d.inch = d.inch % 12;
        d.feet = A.feet + B.feet + d.feet;
        return d;
}
voiddist :: showd()
{
        cout<<feet<<" feet "<<inch<<" inch"<<endl;
```

```
}
int main()
{
        dist d1, d2, d3;
        d1.getd();
        d2.getd();
        d3 = sum(d1,d2);
        cout<<endl;
        d1.showd();
        d2.showd();
        cout<<"----------------------\n";
        d3.showd();
        return 0;
}
```

13. Write a program to create a class time which has data members as hr and min and member function as getd( ) and showd( ) to read and display result. Implement sum( ) as member function to perform addition between two objects. Pass one object as an argument to sum( ).(Perform conversions from inches to feet in function sum).

```
#include<iostream>
usingnamespacestd;
class time
{
        inthr,min;
public:
        voidgetd()
        {
                cout<<"Enter time(in hours and minutes):\n";
                cin>>hr>>min;
        }
        voidshowd()
        {
                cout<<hr<<" : "<<min<<endl;
        }
        time sum(time T)
        {
                time temp;
                temp.min = min+T.min;
                temp.hr = temp.min/60;
                temp.min = temp.min % 60;
                temp.hr = temp.hr+hr+T.hr;
                return temp;
        }
};
int main()
{
        time t1, t2, t3;
        t1.getd();
        t2.getd();
        t3 = t1.sum(t2);
        t1.showd();
        t2.showd();
        cout<<"----------------\n";
        t3.showd();
        return 0;
}
```

14. Write a program to show use of static member function.

```
#include<iostream>
usingnamespacestd;
class bank
{
 staticchar nob[6];
 staticfloatroi;
 int balance;
public:
 voidsetbal()
 {
        cout<<"\nEnter your current balance: ";
        cin>>balance;
 }
 staticvoidsetname()
 {
        strcpy(nob,"ICICI");
 }
 void display()
 {
        cout<<"\nBank Name: "<<nob;
        cout<<"\nBalance amount: "<<balance;
        cout<<"\nRate of interest: "<<roi<<endl;
 }
};
char bank::nob[];
float bank::roi=7.5;

int main()
{
 bank b1,b2;
 bank::setname();
 b1.setbal();
 b1.display();
 b2.setbal();
 b2.display();
 return 0;
}
```

Assignment No. 3

1. Define Constructor. Give syntax.

A constructor is a member function of a class whose task is to initialize objects of a class. In C++,Constructor is automatically called when object(instance of class) is created, it constructs the value of data members of class. A constructor have same name as that of class .It does not have any return type not even void.

There are three types of constructor:

- Default constructor
- Parameterized constructor
- Copy constructor

Syntax:

```
classclass_name
{
Access specifier:
        data members;
        member functions;
public:
        class_name (arguments)
        {
                body of constructor;
        }
};
```

2. Define Destructor. Give syntax.

A destructor is a special member function that is called when lifetime of an object ends. Destructor functions are the inverse of constructor functions. They are called when objects are destroyed (deallocated). Designate a function as a class's destructor by preceding the class name with a tilde (~).

For example, the destructor for class `String` is declared: `~String()`

Syntax:

```
~ class_name ()

{
                body of constructor;
}
```

3. Enlist any four types of constructor.

i. **Default Constructors:** Default constructor is the constructor which doesn't take any argument
ii. **Parameterized Constructors:** Constructor which has arguments is called parameterized constructor.
iii. **Copy Constructor:** A copy constructor is a member function which initializes an object using another object of the same class.
iv. **Dynamic Constructor:** Constructor which allocates the memory to the object at run time is called dynamic constructor.

4. Explain -

i. Parameterized constructor: It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any

other function. When you define the constructor's body, use the parameters to initialize the object.

syntax:

```
class name(parameter1, parameter2..... ,parameter n)
{
        //constructor code;
}
```

ii. Copy constructor: The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously. The copy constructor is used to –

- Initialize one object from another of the same type.
- Copy an object to pass it as an argument to a function.
- Copy an object to return it from a function.

syntax:

```
class name(class_name&obj)
{
        //constructor code;
}
```

5. Write a program to show use of overloading of constructor.

```
#include<iostream>
usingnamespacestd;
class area
{
 floatl,b;
public:
        area()
        {
                l=10;
                b=20;
        }

        area(floatlen, floatbre)
        {
                l=len;
                b=bre;
        }
        floatcal()
        {
                return(l*b);
        }
        voiddisp(float a)
        {
                cout<<"Area of rectangle: "<<a<<endl;
        }
};
int main()
{
 area a1, a2(30,10);
 float a;
 cout<<"Default constructor:"<<endl;
 a=a1.cal();
 a1.disp(a);
```

```
 cout<<"Parameterized constructor:"<<endl;
 a=a2.cal();
 a2.disp(a);
}
```

6. Write a program to show use of destructors. Create any five objects and show their order of destroyable.

```
#include<iostream>
usingnamespacestd;
classabc
{
 staticint count;
public:
        abc()
        {
                count++;
                cout<<"\n Object is created:"<<count<<endl;
        }
        ~abc()
        {

                cout<<"\n Object is destroyed:"<<count<<endl;
                count--;
        }
};
intabc::count;
int main()
{
 cout<<"\nCreation of objects";
 {
        abc s1,s2;
        {
                abc s3,s4,s5;
                cout<<"\nDestruction of objects";
        }
 }
 return 0;
}
```

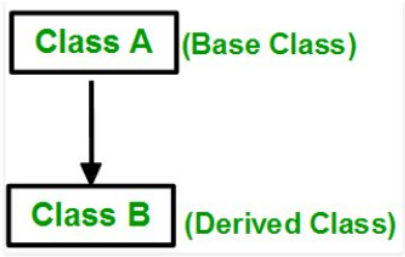
7. Enlist various visibility modes.
   There are 3 types of visibility modes:
   1. Publicinheritance: In this mode, the protected members of super class becomes protected members of sub class and public becomes public.
   2. Privateinheritance: In private mode the protected and public members of super class becomes private member of derived class
   3. Protectedinheritance: In protected mode, the public and protected members of super class becomes protected members of sub class.

8. Define inheritance.
   Inheritance in Object Oriented Programming can be described as a process of creating new classes from existing classes. New classes inherit some of the properties and behaviour of the existing classes. An existing class that is "parent" of a new class is called a base class. Inheritance is a technique of code reuse.
   Syntax:
   classsub_class: access_modesuper_class

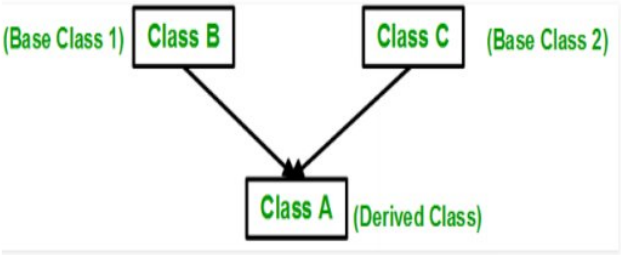
9. Enlist any four types of inheritance.
   1. Single inheritance
   2. Multiple inheritance

3. Multilevel inheritance
4. Hierarchical inheritance
5. Hybrid inheritance

10. Describe four types of inheritance.
    1. Single inheritance - In single inheritance, a class is allowed to inherit from only one class. i.e. one sub class is inherited by one base class only.
       Syntax:
       derived_class_name :acess_modebase_class

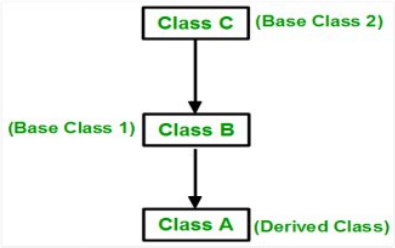


    2. Multiple inheritance – Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.
       Syntax:
       derived_class_name :acess_mode base_class1, acess_mode base_class1....

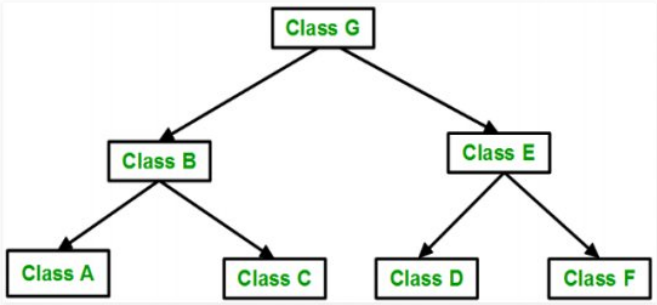


    3. Multilevel inheritance - In this type of inheritance, a derived class is created from another derived class
       The super class for one is subclass for the other.
       Syntax :

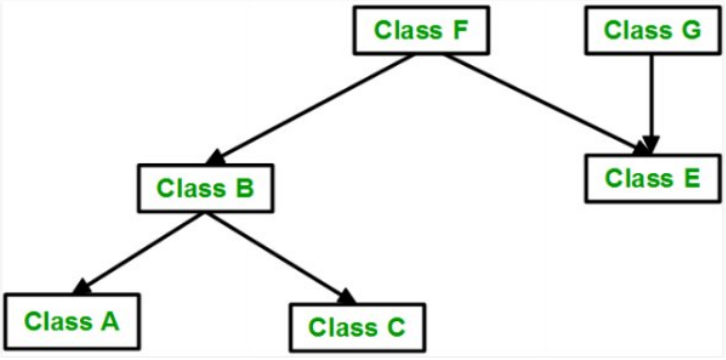
```
class base
{
};
classintermediate_derived : access_mode base
{
};
class derived : access_modeintermediate_derived
{
};
```



4. Hierarchical inheritance - In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class
syntax:
class A
{
};
class B : access_mode A
{
};
class C : access_mode A
{
};



5. Hybrid inheritance - Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.
Below image shows the combination of hierarchical and multiple inheritance:
syntax:



11. Write suitable answer in the following blocks.

Base class	Derived class visibility modes		
	Public	Private	Protected
Public	Public	Private	Protected
Private	NA	Not Accessible	NA
Protected	Protected	Private	Protected

12. Write a program to create a class College which has data members as name and id and member function as cgdata( ) and csdata( ). Derive a class Dept from college which has data member as dname and dcode and member function as dgdata( ) and dsdata( ).Derive a class student from class Dept which has data members sname, srno and member functions as sgdata( ) and ssdata( ).Read and display information for above definition.

```
#include<iostream>
usingnamespacestd;

class college
{
        char name[25];
        int id;
public:
        voidcgdata()
        {
                cout<<"Enter college name and id:\n";
                cin.getline(name,25);
                cin>>id;
        }
        voidcsdata()
        {
                cout<<"\nCollege Name: "<<name;
                cout<<"\nCollege ID: "<<id;
        }
};
classdept:public college
{
        chardname[20];
        intdcode;
public:
        voiddgdata()
        {
                cgdata();
                cout<<"\nEnter department name and code:\n";
                cin.ignore();
                cin.getline(dname,20);
                cin>>dcode;
        }
        voiddsdata()
        {
                csdata();
                cout<<"\nDepartment Name: "<<dname;
                cout<<"\nDepartmnt code: "<<dcode;
        }
};
classstudent:publicdept
{
        charsname[20];
        intsrno;
public:
        voidsgdata()
        {
                dgdata();
                cout<<"\nEnter student name and roll no.:\n";
                cin.ignore();
                cin.getline(sname,20);
                cin>>srno;
        }
        voidssdata()
        {
                dsdata();
                cout<<"\nStudent Name: "<<sname;
                cout<<"\nStudent Roll No.: "<<srno<<endl;
        }
};
```

```
int main()
{
        student S;

        S.sgdata();
        S.ssdata();
        return 0;
}
```

13. Write a program to create a class College which has data members as name and id and member function as cgdata( ) and csdata( ). Derive a class Dept which has data member as dname and dcode and member function as dgdata( ) and dsdata( ).Derive a class student from class College and Dept which has data members sname, srno and member functions as sgdata( ) and ssdata( ).Read and display information for above definition.

```
#include<iostream>
usingnamespacestd;

class college
{
        char name[25];
        int id;
public:
        voidcgdata()
        {
                cout<<"Enter college name and id:\n";
                cin.getline(name,25);
                cin>>id;
        }
        voidcsdata()
        {
                cout<<"\nCollege Name: "<<name;
                cout<<"\nCollege ID: "<<id;
        }
};
Class dept
{
        chardname[20];
        intdcode;
public:
        voiddgdata()
        {
                cout<<"\nEnter department name and code:\n";
                cin.ignore();
                cin.getline(dname,20);
                cin>>dcode;
        }
        voiddsdata()
        {
                cout<<"\nDepartment Name: "<<dname;
                cout<<"\nDepartmnt code: "<<dcode;
        }
};
Class student:public college, public dept
{
        charsname[20];
        intsrno;
public:
        voidsgdata()
```

```
        {
                cout<<"\nEnter student name and roll no.:\n";
                cin.ignore();
                cin.getline(sname,20);
                cin>>srno;
        }
        voidssdata()
        {
                cout<<"\nStudent Name: "<<sname;
                cout<<"\nStudent Roll No.: "<<srno<<endl;
        }
};
int main()
{
        student S;
        S.cgdata();
        S.dgdata();
        S.sgdata();
        S.csdata();
        S.dsdata();
        S.ssdata();
        return 0;
}
```

14. Write a program to create class Employee which has data member as name and id and member function as egdata( ) and esdata( ) .Derive a class Gross_sal from Employee which has data member as bs, da, hra, gs and member function as gdata( ) and cal_sal( ). Derive a class Deduction from class employee which has data member as Ded, PF, LIC and member function as dgdata( )and cal_Ddata(). Derive a class Cal_Netsal( ) to calculate net salary with refrence to above data.
Hint for formulas:
gs=bs+da+hra
Ded=PF+LIC
NS= gs- Ded

```
#include<iostream>
usingnamespacestd;

class employee
{
        char name[20];
        int id;
public:
        voidegdata()
        {
                cout<<"Enter employee name and id:\n";
                cin.getline(name,20);
                cin>>id;
        }
        voidesdata()
        {
                cout<<"\nEmployee Name: "<<name;
                cout<<"\nEmployee ID: "<<id;
        }
};
classGross_sal:virtualpublic employee
{
        floatbs, da, hra;
public:
        floatgs;
        voidgdata()
```

```
        {
                cout<<"\nEnter Basic salary, DA and HRA:\n";
                cin>>bs>>da>>hra;
        }
        voidcal_gsal()
        {
                gs = bs + da + hra;
        }
};
classdeduction:virtualpublic employee
{
        int PF, LIC;
public:
        intDed;
        voiddgdata()
        {
                cout<<"\nEnter PF and LIC:\n";
                cin>>PF>>LIC;
        }
        voidcal_Ddata()
        {
                Ded = PF + LIC;
        }
};
classNet_sal: publicGross_sal,public deduction
{
public:
        intN_sal;
        voidCal_Netsal()
        {
                N_sal = gs - Ded;
                cout<<"\nNet Salary: "<<N_sal;
        }
};
int main()
{
        Net_sal N;
        N.egdata();
        N.gdata();
        N.dgdata();
        N.esdata();
        N.cal_gsal();
        N.cal_Ddata();
        N.Cal_Netsal();
        return 0;
}
```

15. Write a program to pass the parameters from derived class constructor to base class constructor using concept of inheritence.

```
#include<iostream>
usingnamespacestd;

class A
{
 int a;
public:
 A(int x)
 {
        a=x;
 }
 voidasd()
 {
        cout<<"Base:\n"<<a;
 }
};
class B:public A
{
```

```
 intm,n;
public:
 B(intx,inty,int z):A(x)
 {
        m=y;
        n=z;
 }
 voidbsd()
 {
        cout<<"\n\n1st Derived:\n"<<m<<endl<<n;
 }
};
class C: public B
{
 int p;
public:
 C(intl,intm,intn,int o): B(l,m,n)
 {
        p=o;
 }
 voidcsd()
 {
        cout<<"\n\n2nd Derived:\n"<<p<<endl;
 }
};
int main()
{
 C obj(10,20,30,40);
 obj.asd();
 obj.bsd();
 obj.csd();
 return 0;
}
```

16. Write a program to pass the parameters from derived class constructor to base class constructor using concept of multiple inheritence.

```
#include<iostream>
usingnamespacestd;

class A
{
        int a;
public:
        A(int x)
        {
                a=x;
        }
        voidasd()
        {
                cout<<"Base:\n"<<a;
        }
};
class B
{
        intm,n;
public:
        B(inty,int z)
        {
                m=y;
                n=z;
        }
        voidbsd()
        {
                cout<<"\n\n1st Derived:\n"<<m<<endl<<n;
        }
};
class C:public A, public B
{
```

```
        int p;
public:
        C(intl,intm,intn,int o):A(l), B(m,n)
        {
                p=o;
        }
        voidcsd()
        {
                cout<<"\n\n2nd Derived:\n"<<p<<endl;
        }
};
int main()
{
        C obj(10,20,30,40);
        obj.asd();
        obj.bsd();
        obj.csd();
        return 0;
}
```

Assignment No. 4

1. Define this pointer.
   It is predefined variable in every class.
   It is a pointer to the current object i.e. the object whose member is currently being accessed. When a member function is called, this pointer is automatically passed as an implicit argument to the function.
   Syntax:
   this -> member;

2. Enlist any four pointer arithmetic operations.
   - (++) Incrementing a pointer
     Example –
     ```
     int var [3]={11,22,33};
     int *ptr;
     ptr = var;
     ptr ++;
     ```
     We prefer using a pointer in our program instead of an array because the variable pointer can be incremented, unlike the array name which cannot be incremented because it is a constant pointer.
   - (--) Decrementing a pointer
     Example-
     ```
     int var [3]={11,22,33};
     int *ptr;
     ptr = &var[2];
     ptr --;
     ```
     The pointer in this case decreases its value by the number of bytes of its data type
   - (+) Addition of a number to a pointer
     Example –
     ```
     int value = 7;
      int *ptr = &value;
     cout<< ptr+1;
     cout<<ptr+2;
     cout<< ptr+3;
     ```
     In the above example we are adding the number so that the pointer refers to next address.
   - (-) Subtraction of a number from a pointer
     Example –

     ```
     intbuf[10];
     int *p1 = &buf[0];
     int *p2 = 0;
     p1 - p2;
     ```

3. Give any example to create a pointer for the data member. Assume any class.

```
#include<iostream>
usingnamespacestd;
class sample
{
public:
	int a;
	void show()
	{
		cout<<"Value of a : "<<a<<endl;
	}
};
int main()
{
	sample A, *ps;
	ps=&A;
	int sample::*pa=& sample::a;
	ps->pa=10;
	ps->show();
	return 0;
}
```

4. Develop a program to create a class college which has name, code and address as data member and as data member and getdata( ) and showdata( ) as member function to read and display information. Create a pointer for object and using pointer object access member functions.

```
#include<iostream>
usingnamespacestd;
class college
{
	char name[25],address[50];
	int code;
public:
	voidgetd()
	{
		cout<<"Enter name,address and code of college:\n";
		cin.getline(name,25);
		cin.getline(address,50);
		cin>>code;
	}
	voidshowd()
	{
		cout<<"\nName: "<<name<<"\nAddress: "<<address<<"\nCode: "<<code<<endl;
	}
};
int main()
{
	college C,*p;
	p=&C;
	p->getd();
	p->showd();
	return 0;
}
```

5. Write a program to create class “string” and str as character array and member function as getdata( ), cal_sal( ) and showdata( ) to display string and its length. Create pointer for str and using pointer calculate the length of string.

```
#include<iostream>
usingnamespacestd;
class String
{
	charstr[15], *p;
```

```
        int l;
public:
        voidgetdata();
        voidcal_len();
        voidshowdata();
};
void String::getdata()
{
        cout<<"Enter string: ";
        cin.getline(str,15);
}
void String::cal_len()
{
        l=0;
        p=&str[0];
        while(*p!='\0')
        {
                p++;
                l++;
        }
}
void String::showdata()
{
        cout<<"\nLength of String '"<<str<<"' = "<<l<<endl;
}
int main()
{
        String S;
        S.getdata();
        S.cal_len();
        S.showdata();
        return 0 ;
}
```

6. Write a program to create a class “large” and str as integer array and member function as getdata( ) to read integer array, max( ) to find largest element by creating pointer for integer array and showdata( ) to display largest element in array.

```
#include<iostream>
usingnamespacestd;
class large
{
    intstr[10], *p, n, i ,lar;
public:
    voidgetdata();
    void max();
    voidshowdata();
};
void large::getdata()
{
    cout<<"Enter no. of elements in array: ";
    cin>>n;
    cout<<"\nEnter elements in array:\n";
    for(i=0;i<n;i++)
    {
        cin>>str[i];
    }
}
void large::max()
{
    lar=0;
```

```
    p=&str[0];
    for(i=0;i<n;i++)
    {
        if(*p>lar)
        {
            lar=*p;
        }
        p++;
    }
}
void large::showdata()
{
    cout<<"\nLargest element in array is: "<<lar<<endl;
}
int main()
{
    large L;
    L.getdata();
    L.max();
    L.showdata();
    return 0;
}
```

7. Write a program to create a class “string” and srt1, str2 as character array and member function as getdata( ) to read two strings, compare( ) to compare two strings by creating pointer for integer array and showdata( ) to display largest element in array.’

```
#include<iostream>
usingnamespacestd;
class String
{
    char str1[15],str2[15],*s1,*s2;
    int l1,l2,flag;
public:
    voidgetdata()
    {
        cout<<"Enter 2 strings to compare:\n";
        cin.getline(str1,15);
        cin.getline(str2,15);
    }
    void compare();
};
void String::compare()
{
    s1=str1;
    s2=str2;
    l1=strlen(str1);
    l2=strlen(str2);
    flag=0;
    if (l1==l2)
    {
        while(*s1!='\0')
        {
            if(*s1==*s2)
            {
                s1++;
                s2++;
            }
            else
            {
                flag=1;
                break;
```

```
                }
            }
        }
        else
            flag=1;
        if(flag==0)
            cout<<"\nStrings are equal"<<endl;
        else
            cout<<"\nStrings are not equal"<<endl;
}
int main()
{
        String S;
        S.getdata();
        S.compare();
        return 0;
}
```

8. Write a program to create a class “string” and str1, str2 as character array and member function as getdata( ) to read two string, cat ( ) to concatenate two strings by creating pointer for each nd display result.

```
#include<iostream>
usingnamespacestd;
class String
{
        char str1[15],str2[15],*s1,*s2;
public:
        voidgetdata()
        {
                cout<<"Enter 2 strings to concatenate:\n";
                cin.getline(str1,15);
                cin.getline(str2,15);
        }
        void cat();
};
void String::cat()
{
        s1=str1;
        s2=str2;
        while(*s1!='\0')
                s1++;

        while(*s2!='\0')
        {
                *s1=*s2;
                s1++;
                s2++;
        }
        *s1='\0';
        cout<<"\nConcatenated string is: "<<str1<<endl;
}
int main()
{
        String S;
        S.getdata();
        S.cat();
        return 0;
}
```

9. Write a program to create a class “string” and srt1, srt2 as character array and member function as getdata( ) to read str1, reverse ( ) to reverse contents of str1 and store result in str2. Create pointer for str1 and display result.

```
#include<iostream>
usingnamespacestd;
class String
{
	char str1[15], str2[15],*s1;
	int l;
public:
	voidgetdata()
	{
		cout<<"Enter string: ";
		cin.getline(str1,15);
	}
	void reverse();
};
void String::reverse()
{
	l=0;
	s1=&str1[0];
	while(*s1!='\0')
	{
		l++;
		s1++;
	}
	for(int i=0;i<l;i++)
	{
		s1--;
		str2[i] = *s1;
		if(i==l-1)
		{
			str2[i+1]='\0';
		}
	}
	cout<<"\nReverse String: "<<str2<<endl;
}
int main()
{
	String S;
	S.getdata();
	S.reverse();
	return 0;
}
```

10. Write a program to create a class “search” and str as integer array and member function as getdata( ) to read integer array, search( ) to find given element in entered integer array. Create Pointer for integer array and display result whether no found or not.

```
#include<iostream>

usingnamespacestd;
class Search
{
	intstr[10],*p,s,n;
public:
	voidgetdata();
	void search();
};
void Search::getdata()
{
	cout<<"Enter no. of elements in integer array: ";
	cin>>n;
	cout<<"\nEnter elements of integer array:\n";
	for(int i=0;i<n;i++)
	{
		cin>>str[i];
	}
}
void Search::search()
```

```
{
	int flag=1;
	cout<<"\nEnter element to search: ";
	cin>>s;
	p=&str[0];
	for(int i=0;i<n;i++)
	{
		if(*p==s)
		{
			flag=0;
			break;
		}
		p++;
	}
	if(flag==0)
		cout<<"\nNumber is present in the integer array\n";
	else
		cout<<"\nNumber is not present in the integer array\n";

}
int main()
{
	Search S;
	S.getdata();
	S.search();
	return 0;
}
```

Assignment No. 5

1. Define polymorphism.

The word polymorphism is the word made up of POLY + MORPHISM where poly means many and morphism means forms. So, polymorphism means many forms. This usually refers to be able to access different types of objects through a common base class specifically using a pointer of the type of a base object to point to object(s) which derive from that base class. Typically, polymorphism occurs when there is a hierarchy of classes and they are related by inheritance.
In simple words, polymorphism is "one interface, multiple methods".

2. Enlist types of polymorphism.

Types of polymorphism:

A. Compile time polymorphism
   i. Function Overloading
   ii. Operator Overloading
B. Run time polymorphism
   i. Virtual Function

3. i) Give the example to define overloading of + as binary operator outside of the class as normal member function.

```
complexcomplex : : operator +(complex X)
{
        complex temp;
        temp.real = real + X.real;
        temp.imag = imag + X.imag;
        return temp;
}
```

ii) Give the example to define overloading of + as binary operator outside of the class as friend function.

```
complex operator  +(complex A, complex B)
{
        complex temp;
        temp.real = A.real + B.real;
        temp.imag = A.real + B.imag;
        return temp;
}
```

4. Describe

i) Compile time polymorphism –In compile time polymorphism, the compiler is able to select the appropriate function for a particular call at compile time itself. It means that an object is bound to its function call at compile time i.e. linking of function call to its definition at compile time. It is also known as early binding because at the time of compilation only prototype is clucked which cannot be changed and remainsstatic. Function calls are faster as all the information is already available. This can be achieved via 2 ways:
a) Function Overloading
b) Operator Overloading

ii)Run time polymorphism – It means that specifies of appropriate function is done at run time i.e. linking of function call to its definition at run time. This will need the use of a single pointer variable to refer to the objects of different classes. It is also referred as 'dynamic linking' or 'late binding'. Hence, we use the pointer to base class refer to all the derived objects. The compiler ignores the content of the pointer and chooses the member function that matches the type of pointer. For this virtual function are needed.
Run time polymorphism is achieved only when a virtual function is accessed through a pointer to the base class.

5. Differentiate between early binding and late binding.

Early Binding	Late binding
It means that object is bound to its function call at compile time i.e. linking of function call to its definition at compile time.	It means that selection of appropriate function is done at run time i.e. linking of function call to its definition at run time.
Functions to be called are known well before.	Functions to be called is unknown until appropriate selection is made.
It does not require use of pointers to objects.	This requires use of pointers to object.
Function calls are faster.	Function call's execution is slower.
It is also referred as static binding or compile time polymorphism.	It is also referred as late binding or dynamic binding.
Example – Function overloading, operator overloading	Example – Virtual function

6. Describe any 8 rules to overload operators.

1)Only built-in operators can be overloaded. New operators cannot be created.
2) Arity of the operators cannot be changed.
3) Precedence and associativity of the operators cannot be changed.
4) Overloaded operators cannot have default arguments except the function call operator () which can have default arguments.
5) Operators cannot be overloaded for built in types only. At least one operand must be used defined type.
6) Assignment (=), subscript ([]), function call ("()"), and member selection (->) operators must be defined as member functions
7) Except the operators specified in point 6, all other operators can be either member functions or a non-member functions.
8) Some operators like (assignment) =, (address) & and comma (,) are by default overloaded.

7. Describe any 8 rules for virtual function.

1) Only the base class method's declaration needs the virtual keyboard, not the definition.
2) If a function is declared as virtual in the base class, it will be virtual in all its derived classes.
3) Virtual function cannot be static members.
4) It can be friend of another class.
5) The virtual functions are accessed by using object pointer.
6) Virtual constructors cannot be created but virtual destructors can be created.
7) The base pointer can point to any type of the derived object but derived cannot.
8) They must be declared in public section of the class.

8. Enlist the operators that cannot be overloaded.

1) . (Member Access or Dot operator)
2) ?: (Ternary or Conditional Operator)
3) :: (Scope Resolution Operator)
4) .* (Pointer-to-member Operator)

5) sizeof (Object size Operator)

6) typeid (Object type Operator)

9. Enlist the operators that cannot be overloaded as friend function.
   1. Assignment operator =
   2. function call operator ()
   3. subscriping operator []
   4. class member access operator ->

10. Define pure virtual function. Give its syntax.

Pure virtual function are virtual function with no definition. They start with 'virtual' keyword and ends with "=0".
The "=0" is just there to indicate that the virtual function is a pure function and that the function has no body or definition.
Syntax :

```
virtual void function_name()=0;
```

11. Define function overloading.

If any class have multiple functions with same names but different parameters then they are said to be overloaded. Function overloading allows you to use the same name for different functions, to perform, either same or different functions in the same class.
Function overloading is usually used to enhance the readability of the program. If you have to perform one single operation but with different number or types of arguments, then you can simply overload the function

12. Define virtual function.

A virtual function is a member function which is declared within base class and is re-defined (Overridden) by derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call avirtual function for that object and execute the derived class's version of the function.

13. Fill the following blocks with correct answers.

Type of operator to be overloaded	No of arguments to be passed to following functions	
	Member function	Friend function
Unary Operator	0	1
Binary Operator	1	2

14. Write a program to create a class distance which has data member as feet and inch and member function as getd() and showd( ) to read and display result. Overload '+' as friend function to perform addition between two objects. (Perform conversions from inches to feet)

```
#include<iostream>
usingnamespacestd;

classdist
{
        intfeet,inch;
public:
        voidgetd();
```

```
        voidshowd();
        frienddistoperator+(dist, dist);
};
voiddist :: getd()
{
        cout<<"Enter distance (in feet and inches):\n";
        cin>>feet>>inch;
}
distoperator+(dist A, dist B)
{
        dist d;
        d.inch = A.inch + B.inch;
        d.feet = d.inch / 12;
        d.inch = d.inch % 12;
        d.inch = d.inch % 12;
        d.feet = A.feet + B.feet + d.feet;
        return d;
}
voiddist :: showd()
{
        cout<<feet<<" feet "<<inch<<" inch"<<endl;
}
int main()
{
        dist d1, d2, d3;
        d1.getd();
        d2.getd();
        d3 = d1+d2;
        cout<<endl;
        d1.showd();
        d2.showd();
        cout<<"----------------------\n";
        d3.showd();
        return 0;
}
```

15. Write a program to create a class time which has data members as hr and min and member function as getd( ) and showd( ) to read and display result. Overload '+' as member function to perform addition between two objects. (Perform conversions from minutes to hour)

```
#include<iostream>
usingnamespacestd;
class time
{
        inthr,min;
public:
        voidgetd()
        {
                cout<<"Enter time(in hours and minutes):\n";
                cin>>hr>>min;
        }
        voidshowd()
        {
                cout<<hr<<" : "<<min<<endl;
        }
        timeoperator+(time T)
        {
                time temp;
                temp.min = min+T.min;
                temp.hr = temp.min/60;
                temp.min = temp.min % 60;
                temp.hr = temp.hr+hr+T.hr;
                return temp;
        }
};
int main()
{
```

```
        time t1, t2, t3;
        t1.getd();
        t2.getd();
        t3 = t1+t2;
        t1.showd();
        t2.showd();
        cout<<"----------------\n";
        t3.showd();
        return 0;
}
```

16. Write a program to overload '-' as unary operator using concept of friend function.

```
#include<iostream>
usingnamespacestd;
class unary
{
        inta,b;
public:
        unary(intl,int m)
        {
                a=l;
                b=m;
        }
        void display()
        {
                cout<<"a = "<<a<<"\nb = "<<b<<endl<<endl;
        }
        friend unary operator-(unary);
};
unaryoperator-(unary u)
{
        u.a = -u.a;
        u.b = -u.b;
        return u;
}
int main()
{
        unary u1(5,6),u2(-15,-26);
        u1 = -u1;
        u2 = -u2;
        cout<<"Value after implementing '-' sign\n";
        u1.display();
        u2.display();
        return 0;
}
```

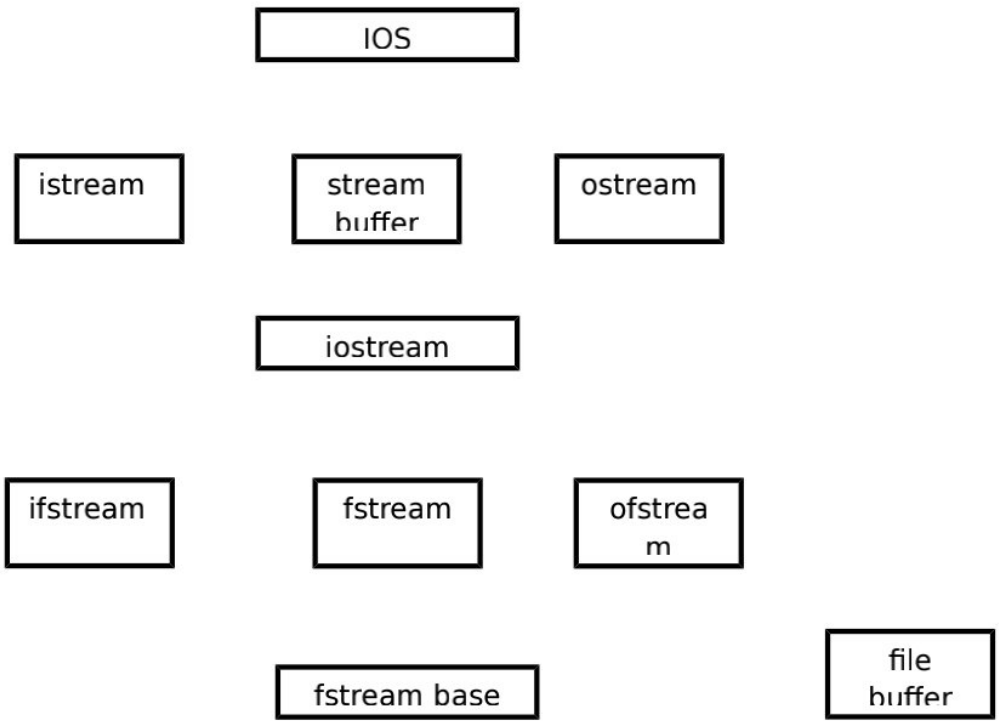
17. Write a program to overload '++' as unary operator using concept of normal member function.

```
#include<iostream>
usingnamespacestd;
class unary
{
        inta,b;
public:
        unary(intl,int m)
        {
                a=l;
                b=m;
        }
        void display()
        {
                cout<<"\na = "<<a<<"\nb = "<<b<<endl;
        }
        voidoperator++();
};
```

```
void unary::operator++()
{
        a++;
        b++;
}
int main()
{
        unary u1(5,6),u2(-9,-11);
        u1++;
        u2++;
        cout<<"Value after increment\n";
        u1.display();
        u2.display();
        return 0;
}
```

Assignment No. 6

1. Draw IOS diagram for file classes.



2. Enlist various operations performed over file.
   I. Writing Operation
   II. Reading Operating
   III. Appending Operation

3. Describe any two methods to open the file with example.

   i. Opening File Using Constructors

   We know that a constructor of class initializes an object of its class when it (the object) is being created. Same way, the constructors of stream classes (ifstream, ofstream, or fstream) are used to initialize file stream objects with the filenames passed to them.

   To open a file named myfile as an input file (i.e., data will be need from it and no other operation like writing or modifying would take place on the file), we shall create a file stream object of input type i.e., ifstream type. Here is an example:ifstreamfin("myfile", ios::in) ;

   The above given statement creates an object, fin, of input file stream. After creating the ifstream object fin, the file myfile is opened and attached to the input stream, fin. Now, both the data being read from myfile has been channelised through the input stream object.

   ii. Opening Files Using Open() Function

   The function open( ) can be used to open multiple files that's use the same stream object.

   For example – we may want to process a set of files sequentially. In such cases, we may create a single stream object and use it to open each file in turn.

   file-stream-class stream-object;
   stream-object.open("filename");

eg-
ofstreamfout;
fout.open("DATA1");

fout.close( );
fout.open("DATA2");

fout.close ( );
the above program segment opens two files in sequence for writing the data.

4. Describe any four file opening modes.
   1) ios : : app
      Append mode. All output to that file to be appended to the end.
   2) ios : : in
      Open a file for reading.
   3) ios : : binary
      This causes a file to be opened in binary mode
   4) ios : : out
      Open a file for writing.
   5) ios : : turnc
      If the file already exists, its contents will be truncated before opening the file.
   6) ios : : ate
      Open a file for output and move the read/write control to the end of the file.

5. Enlist any 8 file opening modes.
   1)ios : : app
   2) ios : : ate
   3) ios : : binary
   4 ios : : in
   5) ios : : nocreate
   6) ios : : noreplace
   7) ios : : out
   8) ios : : turnc

6. Describe any four file errors handling functions.
   1. fail( )
      Returns non-zero value if an operation is unsuccessful. This is carried out by reading the bits ios : : failbit,
      ios : : badbit and ios : : hardfail of ios : : state.
   2. eof( )
      Returns non-zero value when the end of the file is detected; otherwise, it returns zero. This ios : : eofbit is checked.
   3. bad( )
      Returns non-zero value when an error is found in the operation. The ios : : badlist is checked.
   4. good( )
      Returns a non-zero value if no errors occurred during the file operation. This also indicates that the above functions are false. When this function returns true, we can proceed with the file operation.

7. Explain any two methods to detect end of file.
   i. while (fin)
      An ifstream object, such as 'fin', returns a value 0 if any errors occurs in the file operation including the end-of-file condition. Thus, the while loop terminates when fin returns a value of zero on reaching the end of the file condition.
   ii. if(fin.eof( )! = 0) { exit (1); }

eof( ) is a number function of ios class. It returns a non-zero value if the end of file(eof) condition is encountered, and a zero otherwise. Therefore, the above statement terminates the program on reaching the end of file.

8. Describe file pointers with example. OR Explain i)seekg( ) ii)tellg( ) functions.

File pointer –Each file has two associated pointers knows as the file pointers. One of them is called the input pointer (or get pointer) and the other is called the output pointer(or put pointer). We can use these pointers to move through the files while reading or writing The input pointer is used for reading the contents of a given file location and the output pointer is used for writing to a given file location. Each time an input or output operation takes place, the appropriate pointer is automatically advanced.

Example 1 –

the statement
infile.seekg(10);
moves the file pointer to the byte number byte number 10. The bytes in a file are numbered beginning from zero. Therefore, the pointer will be pointing to the 11th byte in the file.

Example 2 –

```
ofstreamfileout ;

fileout.open("hello", ios : : app);

int p=fileout.tellp( );
```

On execution of these statements the output pointer is moved to the end of file "hello" and the value of 'p' will represent the number of bytes in the file.

OR

i. seekg( )

seekg() is a function in the iostream library (part of the standard library) that allows you to seek to an arbitrary position in a file. It is used in file handling to sets the position of the next character to be extracted from the input stream from a given file

For example :

infile.seekg(10);
Moves the file pointer to the byte number 10. The byte in a file are numbered beginning from zero. Therefore, the pointer will be pointing to the 11th byte in the file.

ii. tellg( )

The **tellg()** function is used with input streams, and returns the current "get" position of the pointer in the stream. It has no parameters and returns a value of the member type pos_type, which is an integer data type representing the current position of the get stream pointer.

For example :

```
fin.open ("ABC", ios : : ate)
int size =  fin.tellg( );
```

9. Write a program to write 3 each entries to respective file like "country" and "capital". Display information from above files by reading data from above files and display in following manner on console.
India ---------- New Delhi

America ---------- Washington
England ---------- London

```
#include<iostream>
#include<fstream>
usingnamespacestd;

int main()
{
        ofstream fout1,fout2;
        fout1.open("country");
        fout1<<"India\n";
        fout1<<"America\n";
        fout1<<"England\n";
        fout1.close();

        fout2.open("capital");
        fout2<<"----------New Delhi\n";
        fout2<<"----------Washington\n";
        fout2<<"----------London\n";
        fout2.close();

        ifstream fin1,fin2;
        fin1.open("country");
        fin2.open("capital");
        char data1[20], data2[25];
        while(fin1)
        {
                fin1.getline(data1,20);
                cout<<data1;
                fin2.getline(data2,25);
                cout<<data2;
                cout<<"\n";
        }
        fin1.close();
        fin2.close();
        return 0;
}
```

10. Develop a program to create a class student which has data members as name and roll no. and member function as getd( ) and showd() to read and display information of 10 students. Store the data in file named as “stud_data” and display it on console using read() and write() function.

```
#include<iostream>
#include<fstream>
usingnamespacestd;

class student
{
        char name[20];
        intrno;
public:
        voidgetd()
        {
                cout<<"\nEnter name and roll no. of student:\n";
                cin.ignore();
                cin.getline(name,20);
                cin>>rno;
        }
        voidshowd()
        {
                cout<<"\nName: "<<name;
                cout<<"\nRoll no.: "<<rno<<endl;
        }
```

```
};
int main()
{
        student S[10];
        fstream file("Stud_data",ios::out|ios::in);

        for(int i=0;i<10;i++)
        {
                S[i].getd();
                file.write((char *)&S[i], sizeof(S[i]));
        }
        file.seekg(0);

        for(int j=0;j<10;j++)
        {
                file.read((char *)&S[j], sizeof(S[j]));
                S[j].showd();
        }
        file.close();
        return 0;
}
```

11. Write a program using command line argument to write odd elements to file “ODD” and even elements to file “EVEN” from the entered integer array of 10 elements.

```
#include<iostream>
#include<fstream>
#include<stdlib.h>
usingnamespacestd;
int main(intargc, char * argv[])
{
        intnum[9]={11,22,33,44,55,66,77,88,99};
        if(argc!=3)
        {
                cout<<"argc = "<<argc<<endl;
                cout<<"Error in arguments\n";
                exit(1);
        }
        ofstream fout1, fout2;
        fout1.open(argv[1]);
        if(fout1.fail())
        {
                cout<<"Could not open file"<<argv[1]<<endl;
                exit(1);
        }
        fout2.open(argv[2]);
        if(fout2.fail())
        {
                cout<<"Could not open file"<<argv[2]<<endl;
                exit(1);

        }
        for(int i=0;i<9;i++)
        {
                if(num[i]%2 == 0)
                        fout2<<num[i]<<"  ";
                else
                        fout1<<num[i]<<"  ";
        }
        fout1.close();
        fout2.close();

        ifstream fin;
        char ch;
```

```
        for(int i=1;i<argc;i++)
        {
                fin.open(argv[i]);
                cout<<"Contents of "<<argv[i]<<endl;
                do
                {
                        fin.get(ch);
                        cout<<ch;
                }
                while(fin);
                cout<<"\n\n";
                fin.close();
        }
        return 0;
}
```