1 Example program call by reference
OR
Write a C++ program to swap the values of pair of integers using function and call by reference. (W-16,S-16, S-14, W-14, W-18)

```
#include<iostream>
using namespace std;

void swapptr(int *, int *);
void swapref(int &, int &);

int main()
{
    int a = 45;
    int b = 35;
    cout<<"Before Swap\n";
    cout<<"a="<<a<<" b="<<b<<"\n";

    swapptr(&a,&b);
    cout<<"After Swap with pass by pointer\n";
    cout<<"a="<<a<<" b="<<b<<"\n";

    swapref(a,b);
    cout<<"After Swap with pass by reference\n";
    cout<<"a="<<a<<" b="<<b<<"\n";
    return 0;
}
void swapptr(int *x, int *y)
{
    int z = *x;
    *x=*y;
    *y=z;
}
void swapref(int &x, int &y)
{
    int z = x;
    x = y;
    y = z;
}

Output:
Before Swap
a=45 b=35
After Swap with pass by pointer
a=35 b=45
After Swap with pass by reference
a=45 b=35
```

2 Create two classes X and Y containing private variables x and y respectively. Using a common friend function, perform multiplication operation between x and y. (S-16)

```
#include <iostream>
using namespace std;

// forward declaration
class Y;
class X {
    private:
        int x;
    public:
        X(int a)
        {
            x=a;
        }
      // friend function declaration
      friend int multiply(X, Y);
};

class Y {
    private:
       int y;
    public:
       Y(int b)
        {
            y=b;
        }
       // friend function declaration
       friend int multiply(X, Y);
};

// Function multiply() is the friend function of classes X and Y
// that accesses the member variables x and y
int multiply(X objectX, Y objectY)
{
    return (objectX.x + objectY.y);
}

int main()
{
    X objectX(10);
    Y objectY(20);
    cout<<"Multiplication: "<< multiply(objectX, objectY);
    return 0;
}

Output:
Multiplication: 30
```

3 Write a program to find out sum of two private data members x and y of two classes A and B using a common friend function. Assume that the prototype for both the classes will be void sum (A, B); (W-15)

```
#include <iostream>
using namespace std;

// forward declaration
class B;
class A {
    private:
        int x;
    public:
        A()
        {
            x=12;
        }
        // friend function declaration
        friend int sum(A, B);
};
class B {
    private:
        int y;
    public:
        B()
        {
            y=12;
        }
        // friend function declaration
        friend int sum(A, B);
};
// Function sum() is the friend function of classes A and B
// that accesses the member variables x and y
int sum(A objectA, B objectB)
{
   return (objectA.x + objectB.y);
}

int main()
{
    A objectA;
    B objectB;
    cout<<"Sum: "<< sum(objectA, objectB);
    return 0;
}

Output:
Sum: 24
```

4 Define a class matrix with an integer array of 3X3 as a data member. Define a friend function which adds two matrix objects and returns resultant matrix object. (W-13)

```
#include<iostream>
using namespace std;

class Matrix
{
    int a[3][3];

    public :
    void getMatrix()
    {
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                cin>>a[i][j];
            }
        }
    }

    void displayMatrix()
    {
        for(int i=0;i<3;i++)
        {
            for(int j=0;j<3;j++)
            {
                cout<<a[i][j]<<" ";
            }
            cout<<endl;
        }
    }

    friend Matrix add(Matrix m,Matrix n);
};

Matrix add(Matrix m, Matrix n)
{
    Matrix temp;
    for(int i=0;i<3;i++)
    {
        for(int j=0;j<3;j++)
        {
            temp.a[i][j] = m.a[i][j]+n.a[i][j];
        }
    }
    return temp;
}

int main()
{
    Matrix M1,M2;
    cout<<"Enter values for Matrix M1 : ";
```

```
    M1.getMatrix();
    cout<<"Enter values for Matrix M2 : ";
    M2.getMatrix();

    Matrix M3 = add(M1,M2);

    M1.displayMatrix();
    cout<<endl;
    M2.displayMatrix();
    cout<<"\nSum of Two Matrix : \n";
    M3.displayMatrix();
    return 0;
}
```

```
Output:
Enter values for Matrix M1 : 2 4 5 3 4 5 6 3 2
Enter values for Matrix M1 : 4 1 2 3 4 6 3 1 5
2 4 5
3 4 5
6 3 2

4 1 2
3 4 6
3 1 5

Sum of Two Matrix :
6 5 7
6 8 11
9 4 7
```

5 Write program using Inline Function to find largest of three numbers. (S-13)

```
#include <iostream>
using namespace std;

inline int maximum(int a, int b, int c)
{
    int max;
    max = a > b ? (a > c ? a : c) : (b > c ? b : c);
    return max;
}

int main()
{
    int x, y, z;
    cout<<"Enter values of x, y and z: " ;
    cin>>x>>y>>z;
    cout<<"The maximum number among three is: "<<maximum(x,y,z);
}
```

```
Output:
Enter values of x, y and z: 2 4 6
The maximum number among three is:6
```

6 Create two classes DM and DB which store the value of distances. DM stores distances in meters and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB. Use a friend function to carry out the addition operation. The object stores the results may a DM object or DB object, depending on the units in which the results are required. The display should be in the format of feet and inches or meters and centimeters depending on the object on display.

1 Feet = 0.3048 Meter 1 Meter = 3.28 Feet

1 Inch = 2.54 Centimeter 1 Centimeter = 0.3937 Inch (S-13)

```
#include<iostream>
using namespace std;

class db;
class dm
{
    float metre,cm,tot;
    public:
    void getdm()
    {
        cout << "Enter the distance in meters & centimeters";
        cout << "\nMeters: ";
        cin >> metre;
        cout <<"Centimeters: ";
        cin >> cm;
        tot = metre + 0.01 *(cm);

    }
    friend void add(dm,db);
};

class db
{
    float feet, inches, total;
    public:
    void getdb()
    {
        cout << "\n\nEnter the distance in feets & inches";
        cout << "\nFeet: ";
        cin >> feet;
        cout <<"Inches: ";
        cin >> inches;
        total = feet + 0.0833333 *(inches);
    }
    friend void add(dm,db);
};

//Defining friend function
void add(dm C,db D)
{
    float feets,meter;
    int key;
```

```
    feets = 3.280839*C.tot;
    meter = 0.3048*D.total;
    cout << "In what units do want the addition: "<<endl;
    cout << "1) Meters ";
    cout << "2) Feets ";
    cin>>key;
    if(key==1|| key==2)
    {
        if(key==1) cout << "\n\nMeters: " <<meter+C.tot;
        if(key==2) cout << "\n\nFeets: "<<feets+D.total;
    }
    else cout <<"\n\nIncorrect Entry: PROGRAM WILL EXIT";
}

int main()
{
    dm A;
    db B;

    A.getdm();
    B.getdb();

    add(A,B);
    return 0;
}
```

```
Output:
Enter the distance in meters & centimeters
Meters:
Centimeters:

Enter the distance in feets & inches
Feet:
Inches:

In what units do want the addition:
1)  Meters
2)  Feets

1
Meters: 2.7358
```

7 Explain friend function with the help of an example. (S-17)

```
#include<iostream>
using namespace std;

class Box
{
    double width;
    public:
    void setWidth( double wid )
    {
        width = wid;
    }
    friend void display( Box );
};

void display(Box b)
{
  cout << "Width of box : " << b.width;
}

int main( )
{
    Box b;
    b.setWidth(10.0);
    display(box);
    return 0;
}

Output:
Width of box : 10
```

8 What is overloading of a function? When do we use this concept? Explain with an example. (W-17)

```
#include<iostream>
using namespace std;

float area(int r)
{
  return 3.14*r*r;
}

float area(int h, int b)
{
  return 0.5*h*b;
}

float area(int l, int w, int h)
{
  return l*w*h;
}

int main(){
  cout<<"Area of circle= "<<area(5);
  cout<<"\nArea of triangle= "<<area(4,9);
  cout<<"\nArea of box= "<<area(5,8,2);
  return 0;
}
```

```
Output:
Area of circle= 78.5
Area of triangle= 18
Area of box= 80
```

9 What is a reference variable? What is its major use? Explain with an example. (W-17)

```
#include<iostream>
using namespace std;
void swap(int&,int&);

int main()
{
    int a,b;
    cout<<"Enter value of A: ";
    cin>>a;
    cout<<"Enter value of B: ";
    cin>>b;

    cout<<endl<<"Before function call: A="<<a<<" B="<<b<<endl;
    swap(a,b);
    cout<<"After function call: A="<<a<<" B="<<b<<endl;
    return 0;
}

void swap(int&x,int&y)
{
    int temp;
    temp=x;
    x=y;
    y=temp;

    cout<<"Inside swap function: A="<<x<<" B="<<y<<endl;
}

Output:
Enter value of A: 4
Enter value of B: 9

Before function call: A=4 B=9
Inside swap function: A=9 B=4
After function call: A=9 B=4
```

10 Explain default arguments concept with example. (S-18)

```
#include<iostream>
using namespace std;
int add(int a, int b=5, int c=7)
{
    return (a+b+c);
}

int main()
{
    int x=3;
    cout<<"Addition= "<<add(x);

    return 0;
}

Output:
Addition= 15
```

11 Describe, with examples, the uses of enumeration data types. (S-17)

```
#include<iostream>
using namespace std;

enum month
     {
     january=1,february,march,april,may,june,july,august,september,
     october,november,december
     };

int main()
{
     month currentMonth;
     currentMonth=december;
     cout<<"The Month is December"<<endl;

     //cin not working with Enum

     /* Printing comments on current Month */
     if(currentMonth>=march && currentMonth<=may)
     {
          cout<<"Yay, It is Spring!"<<endl;
     }
     else if(currentMonth>=6 && currentMonth<=august)
     {
          cout<<"It is Summer, Who needs an Ice Cream?"<<endl;
     }
     else if(currentMonth>=september && currentMonth<=november)
     {
          cout << "I am enjoying Autumn, Aren't You?" << endl;
     }
     else
     {
          cout << "Ooh, It is very cold outside! It's Winter!\n";
     }

     return 0;
}

Output:
The Month is December
Ooh, It is very cold outside! It's Winter!
```

12 Write a C++ program to find out the sum and average of three numbers using nesting of member function. (W-18)

```
#include <iostream>
using namespace std;

int sum(int,int,int);
void average(int,int,int);

int main()
{
    int a,b,c;
    cout<<"Enter three numbers: ";
    cin>>a>>b>>c;
    average(a,b,c);

    return 0;
}

int sum(int x,int y,int z)
{
    return (x+y+z);
}

void average(int x,int y,int z)
{
    cout<< "Sum= " << sum(x,y,z)<< endl; //nesting of mem function
    cout << "Average= " << sum(x,y,z)/3.0 << endl;
}

Output:
Enter three numbers: 1 5 7
Sum= 13
Average= 4.33333
```

13 Write a C++ program to find volume of cube, cylinder and rectangular box using concepts of function overloading.(volume of cube = s3, volume of cylinder is p*r2*h, volume of rectangular box is l*b*h) (W-18)

```
#include<iostream>
#include<cstdlib>
using namespace std;

float area(float);
float area(float,float);
float area(float,float,float);

int main()
{
    float l,b,h,s,r;
    int ch;

    do{
    cout<<endl<<"1. Volume of Cube"<<endl;
    cout<<"2. Volume of Cylinder"<<endl;
    cout<<"3. Volume of Rectangle"<<endl;
    cout<<"press 0 to exit"<<endl;
    cout<<"Choose Any: ";

    cin>>ch;
    cout<<endl;

    switch(ch)
    {
        case 0:
            exit(0);
        case 1:
            cout<<"Enter Side of Cube: ";
            cin>>s;
            cout<<"Volume of Cube: "<<area(s)<<endl;
            break;

        case 2:
            cout<<"Enter Radious and height of Cylinder: ";
            cin>>r>>h;
            cout<<"Volume of Cylinder: "<<area(r,h)<<endl;
            break;

        case 3:
            cout<<"Enter Three sides of Rectangle: ";
            cin>>l>>b>>h;
            cout<<"Volume of Rectangle: "<<area(l,b,h)<<endl;
            break;

        default:
            cout<<"Enter Valid Choice"<<endl;
    }
    }while(ch!=0);
```

```
    return 0;
}

float area(float s)
{
    return s*s*s;
}

float area(float r ,float h)
{
    return 3.14*r*r*h;
}

float area(float l,float b,float h)
{
    return l*b*h;
}
```

```
Output:
1. Volume of Cube
2. Volume of Cylinder
3. Volume of Rectangle
press 0 to exit
Choose Any: 2

Enter Radious and height of Cylinder: 2 5
Volume of Cylinder: 62.8

1. Volume of Cube
2. Volume of Cylinder
3. Volume of Rectangle
press 0 to exit
Choose Any: 1

Enter Side of Cube: 3
Volume of Cube: 27

1. Volume of Cube
2. Volume of Cylinder
3. Volume of Rectangle
press 0 to exit
Choose Any: 0
```

14 Consider following class declarations.

```
class ABC;
class XYZ{
        int x;
        public:
            void get_x(int i) {x=i;}
            friend void sum(XYZ,ABC);
        };
class ABC{
        int y;
        public:
             void get_y(int i) {y=i;}
             friend void sum(XYZ,ABC);
        };
```

By considering above class declarations, define a function sum() at outside the class to find out the sum of x and y and display the result. Also define the main() which satisfy above class declarations. (W-18)

```
#include<iostream>
using namespace std;

class ABC;
class XYZ
{
    int x;
    public:
        void get_x(int i)
        {
            x=i;
        }
        friend void sum(XYZ a,ABC b);
};

class ABC
{
    int y;
    public:
        void get_y(int i)
        {
            y=i;
        }
        friend void sum(XYZ a,ABC b);

};

void sum(XYZ a,ABC b)
{
    cout<<endl<<"Sum: "<<a.x+ b.y;
}

int main()
{
    ABC a1;
    XYZ x1;
```

```
    a1.get_y(6);
    x1.get_x(9);

    sum(x1,a1);

    return 0;
}
```

```
Output:
Sum: 15
```

1 Write a program to demonstrate conversion of an object of one class into an object of another class
OR
Program type conversion from class type to basic type and one class type to another class type
OR
Example program conversion function.
OR
Explain the type conversion from basic type to class type and from class type to basic type with proper example.
OR
What is a conversion function? How is it created. Explain with example.
OR
What is type conversion? Explain basic to class and class to class type conversion with example. (W-16, S-16,W-15, S-13, W-13, S-17, W-17, S-18)

Basic type to class type conversion

```
#include<iostream>
using namespace std;

class sample
{
    int a;
    public:
        sample(){}
        sample(int x)
        {
            a=x;
        }
        void disp()
        {
            cout<<"The value of a="<<a;
        }
};
int main()
{
    int m=10;
    sample s;
    s = m; //basic to class type conversion
    s.disp();
    return 0;
}

Output:
The value of a=10
```

Class type to class type conversion

```
#include<iostream>
using namespace std;

class invent2;
```

```
class invent1
{
    private:
        int code;
        int items;
        float price;
    public:
    invent1()
    {
        cout<<"Default Constructor Is Called Of Invent : 1\n\a";
    }
    invent1(int a,int b,float c)
    {
        cout<<"Parameterized Constructor Is Called Of Invent : 1\n\a";
        code=a;
        items=b;
        price=c;
    }
    void putdata()
    {
        cout<<"Code : "<<code<<"\n";
        cout<<"Items : "<<items<<"\n";
        cout<<"Price : "<<price<<"\n";
    }
    int getcode()
    {
        return code;
    }
    int getitems()
    {
        return items;
    }
    float getprice()
    {
        return price;
    }
    operator float()
    {
        return (items*price);
    }
};
class invent2
{
    private:
        int code;
        float value;
    public:
    invent2()
    {
        cout<<"Default Constructor Is Called of Invent : 2\n\a";
        code=0;
        value=0;
    }
```

```
    invert2(float x,float y)
    {
    cout<<"Parameterized Constructor Is Called of Invent ; 2\n\a";
        code=x;
        value=y;
    }
    void putdata()
    {
        cout<<"Code : "<<code<<"\n";
        cout<<"Value : "<<value<<"\n";
    }
    invent2(invent1 p)
    {
        code=p.getcode();
        value=p.getitems()*p.getprice();
    }
};
int main()
{
    invent1 s1(100,5,140.20);
    invent2 d1;
    float total_value;
    total_value = s1; //using operator function
    d1=s1; //using constructor

    cout<<"Product : Details-invent1 Type\n\a";
    s1.putdata();
    cout<<"\nStock Value\n\a";

    cout<<"Product : Details-invent2 Type\n\a";
    d1.putdata();
    return 0;
}

Output:
Parameterized Constructor Is Called Of Invent : 1
Default Constructor Is Called of Invent : 2
Product : Details-invent1 Type
Code : 100
Items : 5
Price : 140.2

Stock Value
Product : Details-invent2 Type
Code : 100
Value : 701
```

2 Create a class coordinate containing x, y and z private variables. Perform operations for incrementing, adding and comparing object(s) by overloading + +, + = and = = operators respectively. Define necessary functions to set and display the variables. (S-16)

```
#include<iostream>
using namespace std;

class coordinate
{
    int x,y,z;
    public:
        coordinate()
        {
            x=y=z=0;
        }
        coordinate(int a,int b,int c)
        {
            x=a;
            y=b;
            z=c;
        }
        void display()
        {
            cout<<"\nx="<<x<<endl;
            cout<<"y="<<y<<endl;
            cout<<"z="<<z<<endl;
        }
        void operator++();
        void operator +=(coordinate);
        int operator==(coordinate);
};

int coordinate::operator ==(coordinate c)
{
    if(x==c.x && y==c.y && z==c.z)
        return 1;
    else
        return 0;
}
void coordinate::operator +=(coordinate c)
{
    coordinate tmp;
    x = x + c.x;
    y = y + c.y;
    z =  z + c.z;
}
void coordinate::operator++()
{
    x++;
    y++;
    z++;
}
```

```
int main()
{
    coordinate c1(7,9,4),c2(5,11,3),c3;

    cout<<"Values of object 1 and 2";
    c1.display();
    c2.display();

    cout<<"After calling ==\n";
    if(c1==c2)
        cout<<"objects are equal\n";
    else
        cout<<"objects are not equal\n";

    cout<<"\nAfter calling +=";
    c3+=c1;
    c3.display();

    cout<<"\nAfter calling ++";
    ++c1;
    c1.display();

    return 0;
}
```

```
Output:
Values of object 1 and 2
x=7
y=9
z=4

x=5
y=11
z=3
After calling ==
objects are not equal

After calling +=
x=7
y=9
z=4

After calling ++
x=8
y=10
z=5
```

3 Write a program to create a class distance containing feet and inches. Using operator keyword, convert an object of class distance into total meters which is a float data type. (1 meter=3.28 feet)
OR
What is an operator function? Explain with an example. (S-16, W-17)

```
#include<iostream>
using namespace std;

class Distance
{
    float feet,inch;
    public:
        Distance()
        {
            feet=0;
            inch=0;
        }
        void get()
        {
            cout<<"Enter Feet : \t";
            cin>>feet;
            cout<<"Enter Inch : \t";
            cin>>inch;
        }

        //Operator Function
        Distance operator +(Distance f)
        {
            Distance sum;
            sum.feet=feet+f.feet;
            sum.inch=inch+f.inch;
            return sum;
        }
        void show()
        {
            cout<<"The Feet Is - \t"<<feet<<endl;
            cout<<"The Inch Is - \t"<<inch<<endl;
        }
};

int main()
{
    cout<<"This Is A Program For Operator Overloading = + \n\a";
    Distance c1,c2,c3;
    c1.get();
    c2.get();
    c3=c1+c2;
    cout<<"Total distance\n";
    c3.show();
    return 0;
}
```

```
Output:
Enter Feet :    1
Enter Inch :    2
Enter Feet :    3
Enter Inch :    4
Total distance
The Feet Is -   4
The Inch Is -   6
```

4 Demonstration of static variables and static functions with a program.
OR
Demonstrate the use of static variables and static functions with a program. (S-16, W-15, S-13, W-14, W-18)

```
#include<iostream>
using namespace std;

class item
{
    int number;
    static int count;// static variable declaration
    public:
    void getdata(int a)
    {
        number = a;
        count++;
    }

   static void getcount() //static method
   {
        cout<<"value of count: \n"<<count;
   }
};

int item :: count; // static variable definition

int main()
{
    item a,b,c;

    a.getdata(100);
    item::getcount(); // call to static method

    b.getdata(200);
    item::getcount();

    c.getdata(300);
    item::getcount();
    return 0;
}

Output:
value of count:1
value of count:2
value of count:3
```

5 Write a program to illustrate the use of copy constructor.
OR
What is constructor? Explain copy constructor with example.
OR
List out the characteristics of constructor in C++. Write a program to illustrate the use of copy constructor. (S-18, W-18)

```
#include<iostream>
using namespace std;

class integer
{
    int m, n;
    public:
        integer(int x,int y)
        {
            m=x;
            n=y;
            cout<<"Constructor Called";
        }

        integer(integer &x)
        {
            m = x.m;
            n = x.n;
            cout<<"\nCopy Constructor Called";
        }
};

int main()
{
    integer i1(5,6);
    integer i2=i1; //invokes copy constructor
    return 0;
}

Output:
Constructor Called
Copy Constructor Called
```

6 Declare a class called book_details to represent details for a book, having data members like title, author, edition, price and no_of_copies_available. Define following functions:
- constructor(s)
- display to display all data members
- find_books to find and display details of all books having price less than Rs. 250
- main to create an array of book_details and to show usage of above functions.

```
#include<iostream>
using namespace std;

class book_details
{
    char title[10], author[10];
    float edition, price;
    public:
        void get_data()
        {
            cout<<"Enter book title=";
            cin>>title;
            cout<<"Enter book author=";
            cin>>author;
            cout<<"Enter book price=";
            cin>>price;
            cout<<"Enter book edition=";
            cin>>edition;
        }
        void display()
        {
            cout<<"Book Title="<<title<<endl;
            cout<<"Book Author="<<author<<endl;
            cout<<"Book Edition="<<edition<<endl;
            cout<<"Book Price="<<price<<endl;
        }
        friend void find(book_details bd[])
        {
            for(int i=0;i<3;i++)
            {
                if(bd[i].price<=250)
                {
                cout<<"Book Title="<<bd[i].title<<endl;
                cout<<"Book Author="<<bd[i].author<<endl;
                cout<<"Book Edition="<<bd[i].edition<<endl;
                cout<<"Book Price="<<bd[i].price<<endl;
                }
            }
        }
};
int main()
{
    book_details bd[3];
    for(int i=0;i<3;i++)
    {
```

```
            bd[i].get_data();
            cout<<endl;
        }
        cout<<"***Displaying books Details***\n";
        for(int i=0;i<3;i++)
        {
            bd[i].display();
            cout<<endl;
        }
        cout<<"\nFollowing books have price less than 250\n";
        find(bd);
        return 0;
    }
```

```
Output:
Enter book title=OOPC++
Enter book author=Balaguru
Enter book price=290
Enter book edition=5

Enter book title=C++
Enter book author=Jarne
Enter book price=190
Enter book edition=2

Enter book title=OOP
Enter book author=Lafore
Enter book price=251
Enter book edition=1

***Displaying books Details***
Book Title=OOPC++
Book Author=Balaguru
Book Edition=290
Book Price=5

Book Title=C++
Book Author=Jarne
Book Edition=190
Book Price=2

Book Title=C++
Book Author=Lafore
Book Edition=251
Book Price=1

Following books have price less than 250
Book Title=C++
Book Author=Jarne
Book Edition=2
Book Price=190
```

7 Declare a class called bird having private data members name and weight. Define following functions :- default constructor for reading data members from key board- overloaded constructor with two arguments to be used for initialization of data members.- display function to display data members.- overloaded member operator >= to compare weight of two bird objects, returning false if weight of first bird object is less than that of the second & true otherwise.

```
#include<iostream>
using namespace std;

class bird
{
    int weight;
    string name;
    public:
        bird(char *nm, int w)
        {
            name = nm;  weight = w;
        }
        void display()
        {
            cout<<"Bird="<<name<<", Weight="<<weight<<endl;
        }
        int operator <= (bird);
};
int bird::operator <= (bird b)
{
    if(weight>=b.weight)
        return 1;
    else
        return 0;
}
int main()
{
    bird b1("eagle",10),b2("parrot",5);
    cout<<"Two birds are\n";
    b1.display();
    b2.display();
    if(b1<=b2)
    {
        cout<<"\nBird-1 has higher weight";
    }
    else
    {
        cout<<"\nBird-2 has higher weight";
    }
    return 0;
}
Output:
Two birds are
Bird=eagle, Weight=10
Bird=parrot, Weight=5
Bird-1 has higher weight
```

8 Declare a class called book having members like book_title, publisher and author_name. Overload extractor and insertion operators (>> and <<) for class book. (S-17)

```
#include <iostream>
using namespace std;

class Book
{
    char title[10], author[10],publisher[10];
    public:
      friend ostream &operator<<( ostream &output, const Book &B )
      {
         output<<"\nTitle : "<< B.title<< "\nAuthor : "<<B.author<<
         "\nPublisher:"<<B.publisher;
         return output;
      }

      friend istream &operator>>( istream  &input, Book &B )
      {
         input >> B.title >> B.author >> B.publisher;
         return input;
      }
};

int main()
{
    Book B1;
    cout<<"Enter the value of Book title, author, publisher: "<<endl;
    cin >> B1;
    cout << "\n---Book Details---" << B1;
    return 0;
}

Output:
Enter the value of Book title, author, publisher :
OOPC++
Balaguru
TATA
---Book Details---
Title : OOPC++
Author : Balaguru
Publisher:TATA
```

9 Define a class complex with real and imaginary as two data member with default & parameterized constructors, function to initialize and display data of class. It should overload the + operator to add two complex objects. Write a complete C++ program to demonstrate use of complex class.

```
#include<iostream>
#include<math.h>
using namespace std;

class complex
{
    private:
        float real,imag;
    public:
        complex( )
        {
            real = imag = 0.0;
        }
        complex(float r, float i)
        {
            real = r;
            imag = i;
        }
        complex(complex &c)
        {
            real = c.real;
            imag = c.imag;
        }

        complex addition(complex d);

        void display()
        {
            cout<<real;
            if(imag < 0)
                cout<<"-i";
            else
                cout<<"+i";

            cout<<fabs(imag);   /* calcutes the absolute value
                                   of a floating point number */
        }
};

complex complex::addition(complex d)
{
    complex temp;
    temp.real=real+d.real;
    temp.imag=imag+d.imag;
    return(temp);
}
```

```
int main( )
{
    complex x1,x2,x3,x4,x5;
    cout<<"\nThe complex numbers in a+ib form :\n\n";
    cout<<"First complex number: ";
    x1.display();

    complex x2(1.5,5.3); //invokes parameterized constructor
    cout<<"\nSecond complex number: ";
    x2.display();

    complex x3(2.4,1.9);
    cout<<"\nThird complex number: ";
    x3.display();

    cout<<"\nAddition of second and third complex number: ";
    x4=x2.addition(x3); //function call
    x4.display( );

    cout<<"\nThe result is copied to another object: ";
    complex x5(x4); //invokes copy constructor
    x5.display();
}

Output:
The complex numbers in a+ib form :

First complex number: 0+i0
Second complex number: 1.5+i5.3
Third complex number: 2.4+i1.9
Addition of second and third complex number: 3.9+i7.2
The result is copied to another object: 3.9+i7.2
```

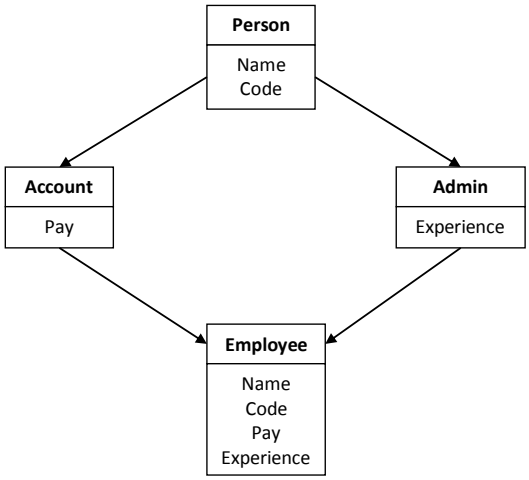
10 Define a class Time with hours and minutes as two data members, add necessary member functions to initialize and display data of class. Do not use constructors in a class. Define a member function sum() which adds two Time objects. Invoke the statements like T3.sum(T1, T2) in main ().

```
#include<iostream>
using namespace std;

class Time
{
    int hours, minutes;
    public:
        void gettime(int h, int m)
        {
            hours =h;
            minutes =m;
        }

        void puttime()
        {
            cout<<hours<<" hours and ";
            cout<<minutes<<" minutes \n";
        }

        void sum(Time t1, Time t2);
};

void Time::sum(Time t1, Time t2)
{
    minutes=t1.minutes+t2.minutes;
    hours=minutes/60;
    minutes=minutes%60;
    hours=hours+t1.hours+t2.hours;
}
int main()
{
    Time T1,T2,T3;
    T1.gettime(4,30);
    T2.gettime(3,37);
    cout<<"T1=";
    T1.puttime();
    cout<<"T2=";
    T2.puttime();
    T3.sum(T1,T2);
    cout<<"T3=";
    T3.puttime();
    return 0;
}

Output:
T1=4 hours and 30 minutes
T2=3 hours and 37 minutes
T3=8 hours and 7 minutes
```

11 Define a class complex with real and imaginary as two data member, add necessary constructors and member function to initialize and display data of class. Class should overload the + operator to add two complex objects and return the results. Invoke the statements like C3=C1+C2 in main ().
OR
Write a C++ program that overloads + operator to add two complex numbers. (S-17, W-18)

```
#include<iostream>
using namespace std;

class complex
{
        int real,imag;
    public:
    complex()
    {
        real=0;
        imag=0;
    }
    complex(int x,int y)
    {
        real=x;
        imag=y;
    }
    void disp()
    {
        cout<<real<<" + "<<imag<<"i"<<endl;
    }
    complex operator + (complex);
};
complex complex::operator + (complex c)
{
    complex tmp;
    tmp.real = real + c.real;
    tmp.imag = imag + c.imag;
    return tmp;
}

int main()
{
    complex c1(4,6),c2(7,9),c3;
    c3 = c1 + c2;
    c1.disp();
    c2.disp();
    c3.disp();
    return 0;
}

Output:
4 + 6i
7 + 9i
11 + 15i
```

12 Consider a class network as shown in figure given below. The class Employee derives information from both Account and Admin classes which in turn derive information from the class Person. Define all the four classes and write a program to create, update and display the information contained in Employee objects.




```
#include<iostream>
using namespace std;

class person
{
    public:
    char name[10];
    int code;

    void getdata()
    {
        cout<<"\n\nEnter name of employee: ";
        cin>>name;
        cout<<"Enter code of employee: ";
        cin>>code;
    }

};

class account: virtual public person
{
    public:
    int pay;

    void getpay()
    {
        cout<<"Enter pay of employee: ";
        cin>>pay;
    }
};
```

```
class admin: virtual public person
{
    public:
    int exp;
    void getexp()
    {
        cout<<"Enter experience of employee: ";
        cin>>exp;
    }
};

class employee: public account, public admin
{
    public:
    void display()
    {
        cout<<"\nName: "<<name;
        cout<<"\nCode: "<<code;
        cout<<"\nExperience: "<<exp;
        cout<<"\nPay: "<<pay;
    }

};

int main()
{
    employee a[3];
    int update_code;
    char choice;

    for(int i=0; i<3;i++)
    {
        a[i].getdata();
        a[i].getpay();
        a[i].getexp();
        a[i].display();
    }

    cout<<"\n\nDo you want to update any employee information y/n? ";
    cin>>choice;

    if(choice=='y')
    {
    out<<"\n\nEnter the code of an employee to update: ";
    cin>>update_code;

        for(int i=0; i<3;i++)
        {
            if(update_code == a[i].code)
            {
                a[i].getdata();
                a[i].getpay();
                a[i].getexp();
```

```
                cout<<"\n\nUpdated information for employee: ";
                a[i].display();
            }
        }
    }
}
```

```
Output:
Enter name of employee: Rohit
Enter code of employee: 1
Enter experience of employee: 2
Enter pay of employee:20000

Name: Rohit
Code: 1
Experience: 2
Pay: 20000

Enter name of employee: Rahul
Enter code of employee: 2
Enter experience of employee: 1
Enter pay of employee: 10000

Name:Rahul
Code:2
Experience:1
Pay: 10000

Enter name of employee: Hiren
Enter code of employee: 3
Enter experience of employee: 3
Enter pay of employee:25000

Name: Ajay
Code: 3
Experience: 3
Pay: 25000

Do you want to update any employee information y/n? y

Enter the code of an employee that you want to update: 1

Enter name of employee: Rohit
Enter code of employee: 1
Enter experience of employee: 3
Enter pay of employee: 25000

Updated information for employee:
Name: Rohit
Code: 1
Experience: 3
Pay: 25000
```

13 Define Class named point which represents 2-D Point, i.e P(x, y). Define Default constructor to initialize both data member value 5, Parameterized constructor to initialize member according to value supplied by user and Copy Constructor. Define Necessary Function and Write a program to test class Point.

```
#include<iostream>
using namespace std;

class point
{
    public:
        int x,y;

        point()
        {
            x=y=5;
        }

        point(int a,int b)
        {
            x=a;
            y=b;
        }

        point(point &c)
        {
            x=c.x;
            y=c.y;
        }

        void display()
        {
            cout<<"P(x,y) = "<<"P("<<x<<","<<y<<")"<<endl;
        }
};

int main()
{
    point p1;
    p1.display();
    point p2(2,2);
    p2.display();
    point p3(p1);
    p3.display();
}

Output:
P(x,y) = P(5,5)
P(x,y) = P(2,2)
P(x,y) = P(5,5)
```

14 Create a class Account. It has three data member account id, name and balance. Define function to assign value and display value. Define function that search account number given by the user. If account number exists, print detail of that account. Write a program using array of object. Declare at least 5 account and print details.

```
#include<iostream>
using namespace std;

class Account
{
    public:
        long double acc_id;
        char name[20];
        double balance;

        void getdata()
        {
            cout<<"\nEnter account id: ";
            cin>>acc_id;
            cout<<"Enter name: ";
            cin>>name;
            cout<<"Enter balance: ";
            cin>>balance;
        }

        void display()
        {
            cout<<"\nAccount id: "<<acc_id;
            cout<<"\nName: "<<name;

            cout<<"\nBalance: "<<balance;
        }
};

int main()
{
    Account a[3];
    long double search;

    for(int i=0; i<5;i++)
    {
        a[i].getdata();
    }

    cout<<"Enter the id of an employee that you want to search: ";
    cin>>search;

    for(int i=0; i<5;i++)
    {
        if(search == a[i].acc_id)
            a[i].display();
    }
}
```

```
Output:
Enter account id: 1
Enter name:Parth
Enter balance:1000
Enter account id: 2
Enter name:Jayesh
Enter balance:2000
Enter account id: 3
Enter name:Mahesh
Enter balance:3000
Enter account id: 4
Enter name:Ajay
Enter balance:4000
Enter account id: 5
Enter name:Pranav
Enter balance:5000

Enter the id of an employee that you want to search: 3

Account id: 3
Name: Mahesh
Balance: 3000
```

15 Define Operator overloading. Create class Time that has three data members hour, minute and second and two constructor, default constructor and parameterized constructor to initialize data member. Write a program to add two times by overloading operator +.
OR
Write a C++ program to find sum of two time quantities in hours and minutes(HH:MM) by overloading the operator +. (S-18)

```
#include<iostream>
using namespace std;

class Time
{
        int hours,minutes,seconds;
    public:
        Time()
        {
            hours=0;
            minutes=0;
            seconds=0;
        }

        void getTime();
        Time operator+(Time);
        void display();
};

void Time :: getTime()
{
    cout<<"Enter Hour(0-11):";
    cin>>hours;
    cout<<"Enter Minutes(0-59):";
    cin>>minutes;
    cout<<"Enter Seconds(0-59):";
    cin>>seconds;
}

Time Time :: operator+(Time t)
{
    Time temp;
    int a,b;

    b=seconds + t.seconds;
    temp.seconds=b%60;

    a=(b/60)+minutes + t.minutes;
    temp.minutes=a%60;

    temp.hours=(a/60)+hours + t.hours;

    return temp;
}
```

```
void Time :: display()
{
    cout<<endl<<"Time: "<<hours<<":"<<minutes<<":"<<seconds;
}

int main()
{
    Time t1,t2,t3;

    cout<<endl<<"Enter First Time: "<<endl;
    t1.getTime();
    cout<<endl<<"Enter Second Time"<<endl;
    t2.getTime();

    t3=t1+t2;
    t1.display();
    t2.display();
    t3.display();

    return 0;
}
```

```
Output:
Enter First Time:
Enter Hour(0-11):5
Enter Minutes(0-59):50
Enter Seconds(0-59):30

Enter Second Time
Enter Hour(0-11):3
Enter Minutes(0-59):40
Enter Seconds(0-59):60

Time: 5:50:30
Time: 3:40:60
Time: 9:31:30
```

16 Write a C++ program to add two complex numbers using friend function. (S-18)

```
#include<iostream>
using namespace std;

class Complex
{
        int real,img;
    public:
        void getData();
        void display();
        friend Complex addData(Complex,Complex);
};

void Complex :: getData()
{
    cout<<"Enter real and imaginary number: ";
    cin>>real>>img;
}

void Complex :: display()
{
    cout<<real <<" + "<<img<<"i"<<endl;
}

Complex addData(Complex c1,Complex c2)
{
    Complex temp;
    temp.real=c1.real+c2.real;
    temp.img=c1.img+c2.img;
    return temp;
}

int main()
{
    Complex c1,c2,c3;
    c1.getData();
    c2.getData();
    c3=addData(c1,c2);
    c1.display();
    c2.display();
    cout<<"--------------"<<endl;
    c3.display();
    return 0;
}
```

```
Output:
Enter real and imaginary number: 2 6
Enter real and imaginary number: 3 7
2 + 6i
3 + 7i
--------------
5 + 13i
```

17 Explain through an example each:
i) Static data member and (ii) Constant member function (S-18)

Static data member

```
#include<iostream>
using namespace std;

class demo
{
        static int count;
    public:
    void getcount()
    {
        cout<<"count="<<++count<<endl;
    }
};
int demo::count;

int main()
{
    demo d1,d2,d3;
    d1.getcount();
    d2.getcount();
    d3.getcount();
    return 0;
}
```

```
Output:
count=1
count=2
count=3
```

Constant member function

```
#include<iostream>
using namespace std;

class Test
{
    int value;
    public:
    Test(int v = 0)
    {
        value = v;
    }

    // We get compiler error if we add a line like "value = 100;"
    // in this function.
    int getValue() const
    {
        return value;
    }
```

```
};

int main()
{
    Test t(20); //or const Test t(20);
    cout<<t.getValue();
    return 0;
}

Output:
20
```

18 How many arguments are required in the definition of and overloaded unary operator? (S-17)

```
#include<iostream>
using namespace std;

class Unaryminus
{
    int n;
    public:
        void getData();
        void operator-();
        void display();
};

void Unaryminus :: getData()
{
    cout<<endl<<"Enter a number: ";
    cin>>n;
}

void Unaryminus :: operator -()
{
    n=-n;
}

void Unaryminus :: display()
{
    cout<<"Unary Minus: "<<n;
}

int main()
{
    Unaryminus p;
    p.getData();
    -p;
    p.display();
    return 0;
}

Output:
Enter a number: 10
Unary Minus: -10
```

---

unary operator using Friend Function

```
#include<iostream>
using namespace std;

class UnaryMinus
{
        int n;
    public:
        void getData();
```

```
        friend void operator-(UnaryMinus&);
        void display();
};

void UnaryMinus :: getData()
{
    cout<<endl<<"Enter a number: ";
    cin>>n;
}

void operator -(UnaryMinus &u) //must pass reference variable
{
    u.n=-u.n;
}

void UnaryMinus :: display()
{
    cout<<"Unary Minus: "<<n;
}

int main()
{
    UnaryMinus p;
    p.getData();
    -p;
    p.display();
    return 0;
}

Output:
Enter a number: 10
Unary Minus: -10
```

19 Following is a main() program.

```
void main()
{
time T1;
int duration = 85;
T1 = duration;
}
```

Where time is a class which contains variables hrs and minutes. duration stores total time in minutes. Define a class time with necessary functions to break up the duration in maximum hours and remaining minutes and store them into hrs and minutes respectively. (W-18)

```
#include<iostream>
using namespace std;

class Time
{
    int hour,minute;
    public :
        Time()
        {
        }

        //Basic to class conversion using constructor
        Time(int x)
        {
            minute=x;
        }

        //class to basic conversion using operator function
        operator int()
        {
            int x;
            x=minute;
            return x;
        }

        void printTime()
        {
            cout<<endl<<"Time: "<<hour<<":"<<minute;
        }

        void addTime(Time t)
        {
            int t1=t;   //class to basic conversion
            minute=t1%60;
            hour=(t-minute)/60;
        }
};
```

```
int main()
{
    Time t1,t2;
    int duration = 85;
    t1 = duration; //basic to class conversion

    t2.addTime(t1);
    t2.printTime();

    return 0;
}

Output:
Time: 1:25
```

1 Virtual base class program and Virtual function program
OR
Explain virtual base class with a program. (W-18)

Virtual function Example

```
#include<iostream>
using namespace std;

class Base
{
    public:
    virtual void show() //virtual function
    {
      cout << "Base\n";
    }
};
class Derv1 : public Base
{
    public:
    void show()
    {
      cout << "Derv1\n";
    }
};
class Derv2 : public Base
{
    public:
    void show()
    {
      cout << "Derv2\n";
    }
};
int main()
{
    Derv1 dv1;
    Derv2 dv2;
    Base* ptr;    //base class pointer
    ptr = &dv1;  //ptr points to derv1
    ptr->show(); //call to derv1 method
    ptr = &dv2;
    ptr->show(); //call to derv2 method
}

Output:
Derv1
Derv2
```

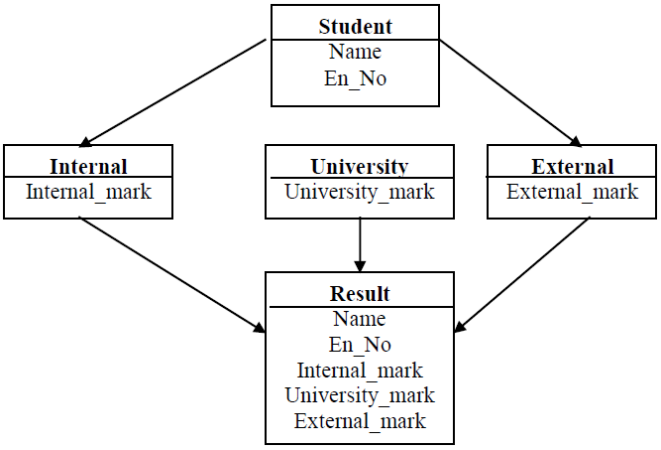
Virtual Class Example

```
#include<iostream>
using namespace std;
class A
{
    public:
        int i;
};

class B:virtual public A
{
    public:
        int j;
};

class C: public virtual A
{
    public:
        int k;
};

class D:public B, public C
{
    public:
        int sum;
};

int main()
{
    D ob1;
    ob1.i=10;
    ob1.j=20;
    ob1.k=30;
    ob1.sum=ob1.i+ob1.j+ob1.k;
    cout<<"Sum = "<<ob1.sum;
}

Output:
Sum = 60
```

2 Hybrid inheritance example

```
#include<iostream>
using namespace std;

class Car
{
    public:
    void display1()
    {
        cout<<"\nCar class";
    }
};

class FuelCar:public Car
{
    public:
    void display2()
    {
        cout<<"\nFuelCar class";
    }
};

class ElecCar:public Car
{
    public:
    void display3()
    {
        cout<<"\nElecCar class";
    }
};

Class HybridCar:public FuelCar, public ElecCar
{
    public:
    void display4()
    {
        cout<<"\nHybridCar class";
    }
};

int main()
{
    Car c; FuelCar f; ElecCar e;
    HybridCar h;
    h.display4();
    h.display3();
    h.display2();
}

Output:
HybridCar class
ElecCar class
FuelCar class
```

3 Consider the following class structure as shown in the figure. The class Result derives information from the classes Internal, University and External respectively. The Internal and External classes access information from the Student class. Define all five classes and write a suitable program to create and display the information contained in Result object.




```
#include<iostream>
using namespace std;

class student
{
        char name[20];
        int en_no;
    public:
    student()
    {
        cout<<"enter name of student:";
        cin>>name;
        cout<<"enter enroll no of student:";
        cin>>en_no;
    }

    char get_n()
    {
        return *name;
    }

    int get_en()
    {
        return en_no;
    }
};

class university
{
        int un_mark;
    public:
    university()
    {
        cout<<"enter marks acquired in university:";
```

```
        cin>>un_mark;
    }

    int get_u_m()
    {
        return un_mark;
    }
};

class internal : virtual public student
{
       int int_mark;
    public:
    internal()
    {
        cout<<"enter internal marks:";
        cin>>int_mark;
    }

    int get_in_m()
    {
        return int_mark;
    }
};

class external : virtual public student
{
       int ext_mark;
    public:
    external()
    {
        cout<<"enter marks got in external:";
        cin>>ext_mark;
    }

    int get_ext_m()
    {
        return ext_mark;
    }
};

class result:public university,public internal,public external
{
    public:
    void display()
    {
        cout<<"\n name of student is:"<<get_n();
        cout<<"\n enroll no is:"<<get_en();
        cout<<"\n marks got in internal is:"<<get_in_m();
        cout<<"\n marks got in external is:"<<get_ext_m();
        cout<<"\n marks got in university is:"<<get_u_m();
    }
};
```

```
int main()
{
    result r;
    r.display();
    return 0;
}
```

```
Output:
enter name of student:shivam
enter enroll no of student:120
enter the marks accuiere in university 55
enter the internal marks 25
enter the marks got in external20

 name of student is:shivam
 enroll no is:120
 marks got in internal is:25
 marks got in external is:20
 marks got in university is:55
```

4 Declare a class called logic_gate to represent logic gates. The class has three data members - input1, input2 and input3 to represent three inputs to the logic gate. The class also has a virtual function member called get_gate_output. Derive two classes from the base class logic_gate, namely, and_gate and or_gate to represent 'logical and gate' and 'logical or gate' respectively. Define function get_gate_output in both of these classes to get the output of the gate. Show use of above classes and functions to demonstrate dynamic polymorphism in function main.

```
#include<iostream>
using namespace std;

class logic_gate
{
        int in_1,in_2,in_3;
    public:
    logic_gate()
    {
        cout<<"enter the values in 1 or 0";
        cout<<"\nenter the first input1:";
        cin>>in_1;
        cout<<"enter the second input2:";
        cin>>in_2;
        cout<<"enter the third input3:";
        cin>>in_3;
    }

    virtual void get_gate_output()=0;

    int in1()
    {
        return in_1;
    }

    int in2()
    {
        return in_2;
    }

    int in3()
    {
        return in_3;
    }
};

class and_gate:public logic_gate
{
        int out;
    public:
    void get_gate_output()
    {
        out=in1()*in2()*in3();
        cout<<"\nAND gate output is:"<<out;
    }
};
```

```
class or_gate:public logic_gate
{
    int out;
    public:
    void get_gate_output()
    {
        out=in1()+in2()+in3();
        if(out>0)
        {
           cout<<"\nOR gate output is = 1";
        }
        else
        {
           cout<<"\nOR gate output is = 0";
        }
    }
};

int main()
{
    and_gate a;
    or_gate o;

    a.get_gate_output();
    o.get_gate_output();

    return 0;
}
```

```
Output:
enter the values in 1 or 0
enter the first input1:1
enter the second input2:0
enter the third input3:1
enter the values in 1 or 0
enter the first input1:1
enter the second input2:0
enter the third input3:1

AND gate output is:0
OR gate output is = 1
```

5 Declare a class called item having data members item_code, item_name, cost and discount. Derive two classes from class item, namely employee and customer. The class employee has data members like employee_code, employee_name and amount. The class customer has data members like customer_name and amount. Define following functions for - initializing data members. - displaying the values of data members. - computing amount to be paid for a purchased item. Also define function main to create objects of both derived classes and to show usage of above functions.

```
#include<iostream>
using namespace std;

class Item
{
    protected:
        int Itemcode;
        char Itemname[50];
        float Cost,Discount;
    public:
        void getData()
        {
            cout<<"Enter Item code:";
            cin>>Itemcode;
            cout<<"Enter Item name:";
            cin>>Itemname;
            cout<<"Enter Cost:";
            cin>>Cost;
            cout<<"Enter Discount:";
            cin>>Discount;
        }
};

class Employee:public Item
{
    private:
        char Emp_name[50];
        int Emp_code;
        float amount;
    public:
        Employee()
        {
            cout<<"Enter Employee name:";
            cin>>Emp_name;
            cout<<"Enter Employee code:";
            cin>>Emp_code;
        }
        void display();
        void calculate();
};

class Customer:public Item
{
    private:
        char Cus_name[50];
        float amount;
```

```
    public:
        Customer()
        {
            cout<<"\nEnter customer name:";
            cin>>Cus_name;
        }
        void display();
        void calculate();
};

void Employee::display()
{
    cout<<"\nEmployee name:"<<Emp_name<<endl;
    cout<<"Employee code:"<<Emp_code<<endl;
    cout<<"Item code:"<<Itemcode<<endl;
    cout<<"Item name:"<<Itemname<<endl;
    cout<<"Cost:"<<Cost<<endl;
    cout<<"Discount:"<<Discount<<endl;
    cout<<"Amount to be paid:"<<amount<<endl;
}

void Employee::calculate()
{
    amount=Cost-Discount;
}

void Customer::calculate()
{
    amount=Cost-Discount;
}

void Customer::display()
{
    cout<<"\nCustomer name:"<<Cus_name<<endl;
    cout<<"Item code:"<<Itemcode<<endl;
    cout<<"Item name:"<<Itemname<<endl;
    cout<<"Cost:"<<Cost<<endl;
    cout<<"Discount:"<<Discount<<endl;
    cout<<"Amount to be paid:"<<amount<<endl;
}

int main()
{
    Employee E1;

    E1.getData();
    E1.calculate();
    E1.display();

    Customer C1;
    C1.getData();
    C1.calculate();
    C1.display();
    return 0;
```

```
}

Output:
Enter Employee name:Tapan
Enter Employee code:009
Enter Item code:113
Enter Item name:mop
Enter Cost:45
Enter Discount:5

Employee name:Tapan
Employee code:9
Item code:113
Item name:mop
Cost:45
Discount:5
Amount to be paid:40
```

6 Create a class student that stores roll_no, name. Create a class test that stores marks obtained in five subjects. Class result derived from student and test contains the total marks and percentage obtained in test. Input and display information of a student.

```
#include<iostream>
using namespace std;

class student
{
        int rn;
        char a[34];
    public:
        void getdata()
        {
            cout<<"Enter the name: ";
            cin>>a;
            cout<<"Enter the Roll no: ";
            cin>>rn;
        }
        void disp_basic()
        {
            cout<<"\nRoll no = "<<rn;
            cout<<"\nName = "<<a;
        }
};

class test
{
    public:
        int m[5];
        void get_marks()
        {
            for(int i=0;i<=4;i++)
            {
                cout<<"Enter mark for subject "<<i+1<<": ";
                cin>>m[i];
            }
        }
};

class result : public student ,public test
{
        int total;
        float per;
    public :
        void count()
        {
            total = 0;
            for(int i=0;i<=4;i++)
            {
                total = total + m[i];
            }
            per = total / 5.0;
```

```
        }

        void disp()
        {
            cout<<"\nTotal Marks = "<<total;
            cout<<"\nPercentage = "<<per;
        }
};

int main()
{
    result s;
    s.getdata();
    s.get_marks();
    s.count();
    s.disp_basic();
    s.disp();
}
```

```
Output:
Enter the name:
Enter the Roll no:
Enter mark for subject 1: 72
Enter mark for subject 2: 74
Enter mark for subject 3: 77
Enter mark for subject 4: 65
Enter mark for subject 5: 82

Roll no = 5
Name = pratik
Total Marks = 370
Percentage = 74
```

7 Assume that Circle is defined using radius and Cylinder is defined using radius and height. Write a Circle class as base class and inherit the Cylinder class from it. Develop classes such that user can compute the area of Circle objects and volume of Cylinder objects. Area of Circle is pie *radius*radius, while volume of Cylinder is pie*(radius * radius)*height.

```
#include<iostream>
using namespace std;
#define pie 3.14

class circle
{
    public:
    float r;
    void getr_forcircle()
    {
        cout<<"Enter the value of radious for circle: ";
        cin>>r;
    }

    void area_circle()
    {
        cout<<"Area of circle = "<<pie*r*r;
    }
};

class cylinder
{
    float h, r;
    public:
        void getdata()
        {
         cout<<"\n\nEnter value of height and radious for cylinder: ";
            cin>>h>>r;
        }

        void area_cylinder()
        {
            cout<<"Area of cylinder = "<<pie*(r*r)*h;
        }

};

int main()
{
    circle c;
    cylinder d;

    c.getr_forcircle();
    c.area_circle();

    d.getdata();
    d.area_cylinder();
}
```

```
Output:
Enter the value of radious for circle: 5
Area of circle = 78.5

Enter value of height and radious for cylinder:3
5
Area of cylinder = 235.5
```

8 When do we use protected visibility specifier to a class member? Explain with an example. (W-17)

```
#include<iostream>
using namespace std;

class Base
{
    private:
        void privateBase()
        {
            cout<<"private Base Function"<<endl;
        }

    protected:
        void protectedBase()
        {
            cout<<"protected Base Function"<<endl;
        }

    public:
        void publicBase()
        {
            cout<<"public Base Function"<<endl;
        }

};

class Derived : public Base
{
    public:
        void derived()
        {
            cout<<"Derived Function"<<endl;
            //privateBase();
            protectedBase();
            publicBase();
        }
};

int main()
{
    Derived d;
    d.derived();

    cout<<"--------------------"<<endl;
    d.publicBase();
    //d.protectedBase();  //Error

    return 0;
}
```

```
Output:
Derived Function
protected Base Function
public Base Function
--------------------
public Base Function
```

9 What are the different forms of inheritance? Give an example for each.
OR
Explain various forms of inheritance with suitable diagrams and example of each. (W-17, W-18)

Single Inheritance

```
#include <iostream>
using namespace std;

//base class
class Vehicle
{
    public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};

//sub class derived from two base classes
class Car: public Vehicle{

};

int main()
{
    // creating object of sub class will
    // invoke the constructor of base classes
    Car obj;
    return 0;
}

Output:
This is a Vehicle
```

Multiple Inheritance

```
#include <iostream>
using namespace std;

//first base class
class Vehicle
{
    public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};

//second base class
class FourWheeler
{
```

```
    public:
    FourWheeler()
    {
      cout << "This is a 4 wheeler Vehicle" << endl;
    }
};

//sub class derived from two base classes
class Car: public Vehicle, public FourWheeler {

};

int main()
{
    Car obj;
    return 0;
}

Output:
This is a Vehicle
This is a 4 wheeler Vehicle
```

Multilevel Inheritance

```
#include <iostream>
using namespace std;

//base class
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};

class fourWheeler: public Vehicle
{  public:
    fourWheeler()
    {
      cout<<"Objects with 4 wheels are vehicles"<<endl;
    }
};

//sub class derived from two base classes
class Car: public fourWheeler
{
   public:
    Car()
    {
        cout<<"Car has 4 Wheels"<<endl;
    }
};
```

```
int main()
{
    Car obj;
    return 0;
}

Output:
This is a Vehicle
Objects with 4 wheels are vehicles
Car has 4 Wheels
```

Hierarchical Inheritance

```
#include <iostream>
using namespace std;

//base class
class Vehicle
{
    public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};


//first sub class
class Car: public Vehicle
{

};

//second sub class
class Bus: public Vehicle
{

};

int main()
{
    Car obj1;
    Bus obj2;
    return 0;
}

Output:
This is a Vehicle
This is a Vehicle
```

Hybrid Inheritance

```
#include <iostream>
using namespace std;
```

```
// base class
class Vehicle
{
  public:
    Vehicle()
    {
      cout << "This is a Vehicle" << endl;
    }
};

//base class
class Fare
{
    public:
    Fare()
    {
        cout<<"Fare of Vehicle\n";
    }
};

//first sub class
class Car: public Vehicle
{

};

//second sub class
class Bus: public Vehicle, public Fare
{

};

int main()
{
    Bus obj2;
    return 0;
}

Output:
This is a Vehicle
Fare of Vehicle
```

1 Example program runtime and compile time polymorphism
OR
What do you mean by dynamic binding? How is it useful in OOP? Explain with an example.
OR
What is polymorphism? Explain with proper example.
OR
Define polymorphism? How is it achieved in C++? Explain with suitable example. (W-17, S-18, W-18)

Runtime polymorphism (Dynamic Binding): (Using Virtual Function)

```
#include<iostream>
using namespace std;
class A
{
    public:
    virtual void show()
    {
        cout<<"Hello base class";
    }
};

class B : public A
{
    public:
    void show()
    {
        cout<<"Hello derive class";
    }
};

int main()
{
    A aobj;
    B bobj;
    A *bptr;
    bptr=&aobj;
    bptr->show();  // call base class function

    bptr=&bobj;
    bptr->show();  // call derive class function
}

Output:
Hello base class
Hello derive class
```

Compile time polymorphism (Static Binding):

Method overloading

```
#include<iostream>
using namespace std;
class Addition
{
    public:
      //function overloading
    void sum(int a, int b)
    {
        cout<<a+b;
    }

      //function overloading
    void sum(int a,int b,int c)
    {
        cout<<a+b+c;
    }
};
int main()
{
    Addition obj;
    obj.sum(10, 20);
    cout<<endl;
    obj.sum(10, 20, 30);
}

Output:
30
60
```

Method overriding

```
#include<iostream>
using namespace std;
class Base
{
 public:
 void show()
 {
  cout<<"Base class\n";
 }
};
class Derived:public Base
{
 public:
 void show()
 {
  cout<<"Derived Class\n";
 }
};

int main()
{
 Base b;   //Base class object
 Derived d;//Derived class object
 b.show(); //Early Binding Ocuurs
 d.show();
}

Output:
Base class
Derived Class
```

2 this pointer with program
OR
Describe the significance of this pointer with suitable example. (W-18)

```
#include<iostream>
using namespace std;

class Test
{
    int mark;
    float spi;
    public:
    void SetData()
    {
        this->mark = 70;
        this->spi = 6.5 ;
    }

    void DisplayData()
    {
        cout << "Mark= "<<mark;
        cout << "spi= "<<spi;
    }
};

int main()
{
    Test o1;
    o1.SetData();
    o1.DisplayData();
    return 0;
}

Output:
Mark= 70, spi= 6.5
```

3 Create a class ITEM with item_code, item_rate and quantity as data members. Create an array of pointers to objects of class ITEM. Write a member function which will calculate the amount of item. Print item_code and amount of item.

```
#include<iostream>
using namespace std;

class item
{
    int code, quantity;
    float price, total_ammount;

    public:
    void getdata(int a, int b, float c)
    {
        code=a;
        quantity=b;
        price=c;
    }

    void show()
    {
        cout<<"Code: "<<code<<endl;
        cout<<"Quantity: "<<quantity<<endl;
        cout<<"Price: "<<price<<endl;
        cout<<"Total amount"<<price*quantity;
    }
};

const int size = 2;
int main()
{
    item *p = new item[size];
    item *d = p;
    int x, z, i;
    float y;

    for(i=0;i<size;i++)
    {
      cout<<"Input code and price and quanitity for item  respectively
      for item "<<i+1<<": ";
        cin>>x>>y>>z;
        p->getdata(x,y,z);
        p++;
    }

    for(i=0; i<size; i++)
    {
        cout<<"\n\nItem: "<<i+1<<endl;
        d->show();
        d++;
    }
    return 0;
}
```

```
Output:
Input code and price and quanitity for item respectively for item 1: 1
3 5
Input code and price and quanitity for item respectively for item 1: 2
5 2

Item: 1
Code: 1
Quantity: 3
Price: 5
Total amount: 15

Item: 2
Code: 2
Quantity: 5
Price: 2
Total amount: 10
```

4 Explain with an example, how would you create a space for an array of objects using pointers. (W-17)

```
#include<iostream>
using namespace std;

class Student
{
        int rno;
        char name[20];
    public:
        void getdata()
        {
        cout<<"Enter Roll number and Name: ";
            cin>>rno;
            cin.getline(name,20);
        }

        void display()
        {
            cout<<endl<<"Roll number: "<<rno;
            cout<<endl<<"Name: "<<name;
        }
};

int main()
{
    const int size=2;
    Student *ptr=new Student[size]; //similar to *ptr=&d[0]
    Student *nptr=ptr;

    for(int i=0;i<size;i++)
    {
        ptr->getdata();
        ptr++;
    }

    for(int i=0;i<size;i++)
    {
        nptr->display();
        nptr++;
    }
}

Output:
Enter Roll number and Name: 1 darshan
Enter Roll number and Name: 2 raj

Roll number: 1
Name:  darshan
Roll number: 2
Name:  raj
```

5 **Create a base class called 'SHAPE' having two data members of type double, member function get_data( ) to initialize base class data members, pure virtual member function display_area( ) to compute and display the area of the geometrical object. Derive two specific classes 'TRIANGLE' and 'RECTANGLE' from the base class. Using these three classes design a program that will accept dimension of a triangle / rectangle interactively and display the area.**
OR
Explain pure virtual function with suitable example. (S-18, W-18)

```
#include<iostream>
#include<fstream>
using namespace std;

class Shape
{
    protected:
        double h,w;
    public:
        void get_data(double h, double w)
        {
            this->h=h;
            this->w=w;
        }
        virtual void display_area() =0;
};

class Rectangle : public Shape
{
    void display_area()
    {
        cout<<"Area of Rectangle: "<< h* w<<endl;
    }
};

class Triangle : public Shape
{
    void display_area()
    {
        cout<<"Area of Triangle: "<< 0.5*h*w<<endl;
    }
};

int main()
{
    Shape *p;
    Triangle t;
    Rectangle r;

    p=&r;
    p->get_data(10,20);
    p->display_area();

    p=&t;
    p->get_data(10,20);
```

```
    p->display_area();
    return 0;
}
```

```
Output:
Area of Rectangle: 200
Area of Triangle: 100
```

6 Explain pure virtual function with suitable example. (W-18)

```
#include<iostream>
using namespace std;

class Base
{
    public:
        virtual void display()
        {
            cout<<"Base class"<<endl;
        }
};

class Derived : public Base
{
    public:
        void display()
        {
            cout<<"Derived class"<<endl;
        }
};


int main()
{
    Base b,*bptr;
    Derived d,*dptr;

    bptr = &b;
    bptr->display(); //Base Class

    dptr = &d;
    dptr->display (); //Derive Class

    bptr=&d;
    bptr->display(); //Derive Class
    return 0;

    /* Here, display() of base class should be declared as virtual
    otherwise it will call display() of Base class instead of Derive
    Class. */
}

Output:
Base class
Derived class
Derived class
```

1 Write a program that reads the information like Name, age and weight of ten people from the keyboard and displays it on the screen in four columns: Sr. no, Name, Weight and Age. Strings should be left-justified and numbers should be right-justified in a suitable field width.

```
#include <iostream>
using namespace std;
class people
{
    char name[20];
    int age,weight;
    public:
    void getdata()
    {
        cout<<"Enter name=";
        cin>>name;
        cout<<"Enter age=";
        cin>>age;
        cout<<"Enter weight=";
        cin>>weight;
    }

    void displaydata()
    {
        cout.width(9);
        cout << std::left << name;
        cout.width(9);
        cout << std::right << weight;
        cout.width(9);
        cout << std::right << age<<endl;
    }
};

int main()
{
    people p[10];
    int count=0;

    for(int i=0;i<10;i++)
    {
        p[i].getdata();
    }

    cout.width(9);
    cout << std::left << "Sr.no";

    cout.width(9);
    cout<<std::left<<"Name";

    cout.width(9);
    cout<<std::right<<"Weight";

    cout.width(9);
    cout<<std::right<<"Age"<<endl;
```

```
    for(int i=0;i<10;i++)
    {
        std::cout.width(9);
        std::cout << std::left << ++count;
        p[i].displaydata();
    }
}
```

```
Output:
Enter name=aakash
Enter age=20
Enter weight=60
Enter name=tapan
Enter age=21
Enter weight=65
Enter name=aarti
Enter age=20
Enter weight=45
...
...
Sr.no     Name          Weight       Age
1         aakash            60        20
2         tapan             65        21
3         aarti             45        20
...
...
```

2 A file contains a list of telephone numbers in the following form: John 23406Ahmed 56789... ...The names contain only one word and the names and telephone numbers are separated by white spaces. Write a program to read the file and output the list in two columns. The names should be left-justified and numbers should be right-justified in a suitable field width.

```
#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

int main()
{
    string name;
    int telephone;
    fstream file1;

    file1.open("list.txt",ios::in|ios::out);
    while(!file1.eof())
    {
        file1>>name;
        file1>>telephone;
        cout << std::left  << setw(30) << name;
        cout << std::right << setw(12) << telephone << endl;
    }
    return 0;
}
```

```
Output:
John                                23456
Ahmed                                9876
Joe                                  4568
```

3 Write a program which uses command line argument to copy the contents of a file A.txt into another file B.txt by reversing case of the characters. E.g. File A.txt: aBCd File B.txt: AbcD

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
    fstream file1, file2;
    char ch;
    file1.open("file1.txt",ios :: out);
    cout << "\n\t Press z and 'enter key' to quit \n";
    cout << "\nEnter the data \n";
    cin.get(ch);

    while(ch != 'z')
    {
        file1.put(ch);
        cin.get(ch);
    }
    file1.close();

    file1.open("file1.txt",ios :: in);
    cout << "\nFirst file contents \n";

    while(file1)
    {
        file1.get(ch);
        cout<<ch;
    }
    file1.close();

    file1.open("file1.txt",ios::in);
    file2.open("file2.txt",ios :: out );

    cout<<"First file contents copied to second file..\n";
    while(file1)
    {
        file1.get(ch);
        if(ch>=65 && ch<=92)
        {
            ch=ch+32;
        }
        else if(ch>=97 && ch<=124)
        {
            ch=ch-32;
        }
        file2.put(ch);
    }
    file2.close();
```

```
    file2.open("file2.txt",ios :: in);
    cout << "\nSecond file contents \n";
    while(file2)
    {
        file2.get(ch);
        cout<<ch;
    }
    file2.close();
    return 0;
}
```

```
Output:
         Press z and 'enter key' to quit

Enter the data
this is india
z

First file contents
this is india

First file contents copied to second file..

Second file contents
THIS IS INDIA
```

4 Explain various file mode parameters in C++. Write a program to copy the contents of a source file student1.txt to a destination file student2.txt character by character.
OR
Write a C++ program to copy the contents of a file A.txt into another file B.txt. (S-17, W-18)

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
    fstream file1, file2;
    char ch;

    file1.open("student1.txt",ios :: out);

    cout << "\n\t Press z and 'enter key' to quit \n";
    cout << "\nEnter the data \n";
    cin.get(ch);

    while(ch != 'z')
    {
        file1.put(ch);
        cin.get(ch);
    }
    file1.close();

    file1.open("student1.txt",ios :: in);
    cout << "\nFirst file contents \n";

    while(file1)
    {
        file1.get(ch);
        cout<<ch;
    }
    file1.close();

    file1.open("student1.txt",ios::in);
    file2.open("student2.txt",ios :: out );

    cout << "\nFirst file contents copied to second file..\n";
    while(file1)
    {
        file1.get(ch);
        file2.put(ch);
    }
    file2.close();

    file2.open("student2.txt",ios :: in);
    cout << "\nSecond file contents \n";
    while(file2)
    {
        file2.get(ch);
```

```
        cout<<ch;
    }
    file2.close();
    return 0;
}
```

```
Output:
        Press z and 'enter key' to quit


Enter the data
this is student file
z

First file contents
this is student file

First file contents copied to second file

Second file contents
this is student file
```

5 Write a program that opens two text files for reading data. It creates a third file that contains the text of first file and then that of second file (text of second file to be appended after text of the first file, to produce the third file).

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
    char ch1,ch2;

    fstream file1,file2;
    fstream file3;

    file1.open("one.txt", ios::in );
    file2.open("two.txt",ios::in);
    file3.open("three.txt",ios::app);

    while(!file1.eof())
    {
        file1.get(ch1);
        cout<<ch1;
        file3.put(ch1);
    }
    file1.close();

    while(!file2.eof())
    {
        file2.get(ch2);
        cout<<ch2;
        file3.put(ch2);
    }

    file2.close();
    file3.close();

    return 0;
}

Output:
hello there. this is file one
another line of file one..hello there. this is file two
another line of file two..
```

6 Write a program that reads a text file and creates another text file that is identical except that every letter must be converted to lower case irrespective of its original case (e.g 'a' or 'A' will become 'a').

```
#include<iostream>
#include<fstream>
#include<string.h>
using namespace std;

int main()
{
    char c[100];

    ofstream f1("File1.txt");
    cout<<"Enter String : ";
    cin.getline(c,100);
    f1<<c;
    f1.close();

    ifstream f2("File1.txt");
    f2.getline(c,50);
    strlwr(c);
    cout<<"Lowercase : "<<c<<endl;
    f2.close();

    ofstream f3("File2.txt");
    f3<<c;
    f3.close();
}

Output:
Enter String : DIET
Lowercase : diet
```

7 Write a program that reads a text file and creates another file that is identical except that every character is in upper case.

```
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    fstream file1, file2;
    char ch;

    file1.open("file1.txt",ios :: out);
    cout << "\n\t Press z and 'enter key' to quit \n\n";
    cout << "\nEnter the data \n";
    cin.get(ch);

    while(ch != 'z')
    {
        file1.put(ch);
        cin.get(ch);
    }
    file1.close();

    file1.open("file1.txt",ios :: in);
    cout << "\nFirst file contents \n";

    while(file1)
    {
         file1.get(ch);
         cout<<ch;
    }
    file1.close();

    file1.open("file1.txt",ios::in);

    file2.open("file2.txt",ios :: out );
    cout<<"\nFirst file contents copied to second file... \n";
    while(file1)
    {
         file1.get(ch);
         if(ch>=97 && ch<=124)
         {
             file2.put(ch);
         }

    }
    file2.close();

    file2.open("file2.txt",ios :: in);
    cout << "\nSecond file contents \n";


    while(file2)
```

```
    {
        file2.get(ch);
        cout<<ch;
    }
    file2.close();

    return 0;
}
```

```
Output:
Enter the data
todayISholiday

First file contents
todayISholiday

First file contents copied to second file...

Second file contents
Todayholidayy
```

8 Write a program that reads a text file and creates another file that is identical except that every sequence of consecutive blank space is replace by a single space. (W-17)

```
#include<fstream>
using namespace std;
int main()
{
    fstream file1, file2;
    char ch;
    int count;

    file1.open("file1.txt",ios :: out);
    cout << "\n\t Press z and 'enter key' to quit \n\n";
    cout << "\nEnter the data \n";
    cin.get(ch);

    while(ch != 'z')
    {
        file1.put(ch);
        cin.get(ch);
    }
    file1.close();

    file1.open("file1.txt",ios :: in);
    cout << "\nFirst file contents \n";

    while(file1)
    {
        file1.get(ch);
        cout<<ch;
    }
    file1.close();

    file1.open("file1.txt",ios::in);
    file2.open("file2.txt",ios :: out );
    cout<<"\nFirst file contents copied to second file... \n";

    while(!file1.eof())
    {
        ch = (char)file1.get( ); //hhh   www
        if(isspace(ch))
            count++;
        if(count >= 2)
        {
            ch=' ';
            count = 0;
        }
        else
        {
            file2 <<ch;
        }

    }
```

```
    file1.close();
    file2.close();

    file2.open("file2.txt",ios :: in);
    cout << "\nSecond file contents \n";

    while(!file2.eof())
    {
        file2.get(ch);
        cout<<ch;
    }
    file2.close();

    return 0;
}
```

```
Output:
Enter the data
hello  world
z

First file contents
hello  world

First file contents copied to second file...

Second file contents
hello world
```

9 Write a C++ program to write text in the file. Read the text from the file from end of file. Display the contents of the file in reverse order. (S-18)

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
    char ch;
    int loc;
    char str[20];

    fstream file1;
    ofstream file2;

    file1.open("diet.txt",ios::out | ios::in);
    cout<<"Enter Content to file1 : ";

    cin.getline(str,50);
    file1<<str;
    file1.close();

      //ate-to set pointer at end of file
    file1.open("diet.txt",ios::in |ios::ate );
    file2.open("revDiet.txt",ios::out);

      //file1.seekg(0,ios::end);
    loc=file1.tellg();
    cout<<endl<<"pointer of file2 at location "<<loc<<endl;
    loc=loc-1;

    while(loc>=0)
    {
        file1.seekg(loc);
        file1.get(ch);
        cout<<ch;
        file2.put(ch);
        loc--;
    }
    cout<<endl<<"file copied in reverse order succesfully";

    file1.close();
    file2.close();
    return 0;
}

Output:
Enter Content to file1 : Darshan Institute of Engg and Tech

pointer of file2 at location 34
hceT dna ggnE fo etutitsnI nahsraD
file copied in reverse order succesfully
```

10 Desribe different ways to open file with suitable example. (W-18)

```
#include<iostream>
#include<fstream>
using namespace std;

int main()
{
    //File Opening using Constructor
    fstream file1("n.txt", ios::out | ios::in );
    if(!file1.is_open())
        cout<<"Error opening file";
    else
    {
        cout<<"successfully opened file1"<<endl;
    }
    file1.close();

    //File Opening using open() function
    fstream file2;
    file2.open("n.txt", ios::out | ios::in );
    if(!file2.is_open())
        cout<<"Error opening file";
    else
    {
        cout<<"successfully opened file2"<<endl;
    }
    file2.close();
    return 0;
}

Output:
successfully opened file1
successfully opened file2
```

1 Example program multiple catch statements. OR Multiple handlers in exception handling program
OR
Define Exception? Explain Exception Handling Mechanism. Write a program that demonstrates use of multiple catch. (W-18)

```
#include<iostream>
using namespace std;
void test(int x)
{
    try
    {
        if(x==1)
            throw x;

        else if(x==0)
            throw 'x';

        else if(x==-1)
            throw 5.14;
    }

    catch(int i)
    {
      cout<<"\nCaught an integer";
    }

    catch(char ch)
    {
      cout<<"\nCaught a character";
    }

    catch(double i)
    {
      cout<<"\nCaught a double";
    }
};

int main()
{
    test(1);
    test(0);
    test(-1);
}

Output:
Caught an integer
Caught a character
Caught a double
```

2 Write a function template for finding the minimum value contained in an array. (W-17)

```
#include<iostream>
using namespace std;

template<typename T>

T minimum(T a[],int size) //template function
{
    T min=a[0];
    for(int i=0;i<size;i++)
    {
        if(a[i]<min)
        min=a[i];
    }
    return min;
}

int main()
{
    int a[10],size,i,min1;
    float b[10],min2;

    cout<<"enter the size value:\n";
    cin>>size;

    cout<<"enter the integer array elements\n";
    for(i=0;i<size;i++)
    {
        cin>>a[i];
    }

    cout<<"enter the floating array elements\n";
    for(i=0;i<size;i++)
    {
        cin>>b[i];
    }

    min1=minimum(a,size);
    min2=minimum(b,size);

    cout<<"The minimum integer elements is:\n";
    cout<<min1;
    cout<<"\nThe minimum floating elements is :\n";
    cout<<min2;

    return 0;
}
```

```
Output:
enter the size value:
5
enter the integer array elements
3
6
7
8
9
enter the floating array elements
9.5
6.2
3.7
11.3
2.1
The minimum integer elements is:
3
The minimum floating elements is :
2.1
```

3 Create a generic class stack using template and implement common Push and Pop operations for different data types.

```
#include <iostream>
#include <string>
using namespace std;

template <class T>
class Stack
{
    public:
        Stack();
        void push(T i);
        T pop();
    private:
        int top;
        T st[100];
};

template <class T>
Stack<T>::Stack()
{
    top = -1;
}
template <class T>
void Stack<T>::push(T i)
{
    st[++top] = i;
}
template <class T>
T Stack<T>::pop()
{
    return st[top--];
}

int main ()
{
    Stack<int> int_stack;
    Stack<string> str_stack;
    Stack<float> float_stack;

    int_stack.push(10);
    float_stack.push(5.5);
    str_stack.push("World");

    cout << int_stack.pop() << endl;
    cout << float_stack.pop() << endl;
    cout << str_stack.pop() << endl;
}

Output:
10
5.5
World
```

4 Program to handle Exception "division by zero" situation.
OR
What is an exception? What are the advantages of using exception handling in a program? Illustrate C++ exception handling mechanism.
OR
Explain try, catch and throw. Give one simple example. (S-17, S-18)

```
#include<iostream>
using namespace std;

int main()
{
    int a,b,c;

    cout<<"Enter value a=";
    cin>>a;
    cout<<"Enter value b=";
    cin>>b;
    try
    {
        if(b!=0)
        {
            c=a/b;
            cout<<"answer="<<c;
        }
        else
        {
            throw(b);
        }
    }
    catch(int x)
    {
        cout<<"Exception caught: Divide by zero\n";
    }
}

Output:
Enter value a=5
Enter value b=0
Exception caught: Divide by zero
```

5 **Declare a template class called exam having an array of generic type as a data member, named elements[5]. Define following generic (template) member functions: - sort to arrange elements in ascending order- find_max to find and return maximum from the array Define main to illustrate usage of these functions to process two different types of data.**
OR
What are class templates? What is the need for class templates? Create a template for bubble sort function.(S-18)

```
#include<iostream>
using namespace std;
template<class exam>

class temp
{
    exam elements[5];
    int size;
    public:
    void sort(exam arr[])
    {
        for(int i=0;i<4;i++)
        {
            for(int j=0;j<4-i;j++)
            {
                if(arr[j]>arr[j+1])
                {
                    exam tmp;
                    tmp=arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=tmp;
                }
            }
        }
    }

    exam maximum(exam arr[])
    {
        exam max=arr[0];
        for(int i=1;i<5;i++)
        {
            if(arr[i]>max)
            {
                max=arr[i];
            }
        }
        return max;
    }
};

int main()
{
    temp<int>t1;
    temp<float>t2;
    int arr1[5];
```

```
    float arr2[5];

    cout<<"Enter 5 interger values:";
    for(int i=0;i<5;i++)
    {
        cin>>arr1[i];
    }
    cout<<"Enter 5 float values:";
    for(int i=0;i<5;i++)
    {
        cin>>arr2[i];
    }

    t1.sort(arr1);
    t2.sort(arr2);

    cout<<"Maximum Interger="<<t1.maximum(arr1)<<endl;
    cout<<"Maximum Float="<<t2.maximum(arr2)<<endl;

    cout<<"\nsorted interger values\n";
    for(int i=0;i<5;i++)
    {
        cout<<arr1[i]<<endl;
    }

    cout<<"\nsorted float values\n";
    for(int i=0;i<5;i++)
    {
        cout<<arr2[i]<<endl;
    }

    return 0;
}

Output:
Enter 5 interger values: 9 2 6 1 8
Enter 5 float values: 10.5  2.6  8.3  20.4  20.1
Maximum Interger=9
Maximum Float=20.4

sorted interger values
1
2
6
8
9

sorted float values
2.6
8.3
10.5
20.1
20.4
```

6 Write program to swap Number using Function Template. Function prototype is given below: void swap(int, int , float , float)Swap two integer number and swap two float number.

```
#include <iostream>
using namespace std;
template <typename T>

void Swap(T &n1, T &n2)
{
    T temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}

int main()
{
    int i1=1, i2=2;
    float f1=1.1, f2=2.2;
    cout<<"Before passing data to function template.\n";
    cout<<"i1 = "<<i1<<"\ni2 = "<<i2;
    cout<<"\nf1 = "<<f1<<"\nf2 = "<<f2;

    Swap(i1, i2);
    Swap(f1, f2);

      cout<<"\n\nAfter passing data to function template.\n";
    cout<<"i1 = "<<i1<<"\ni2 = "<<i2;
    cout<<"\nf1 = "<<f1<<"\nf2 = "<<f2;
    return 0;
}

Output:
Before passing data to function template.
i1 = 1
i2 = 2
f1 = 1.1
f1 = 2.2

After passing data to function template.
i1 = 2
i2 = 1
f1 = 2.2
f1 = 1.1
```

7 By giving an example, illustrate use and working of nested try blocks and re - throwing of an exception. (S-17)

```
#include<iostream>
using namespace std;

int main()
{
    int a=1;

    try
    {
        try
        {
            throw a;
        }
        catch(int x)
        {
            cout<<"\nException in inner try-catch block.";
            throw;
        }
    }
    catch(int n)
    {
        cout<<"\nException in outer try-catch block.";
    }

    cout<<"\nEnd of program.";
    return 0;
}
```

```
Output:
Exception in inner try-catch block.
Exception in outer try-catch block.
End of program.
```

8 Write a class template to represent a generic vector. Include member function to create the vector and to modify the value of a given element. (W-17)

```
#include<iostream>
using namespace std;
template<class T>

class vector
{
        int size;
        T *v;
    public:
        vector();
        void modify();
        void display();
};

template<class T>
vector<T>::vector()
{
    cout<<"\nEnter Number of Co-ordinates : ";
    cin>>size;
    v=new T[size];

    cout<<"\nEnter " << size << " Co-ordinates : \n";
    for(int i=0; i<size; i++)
    {
        cout<<" ";
        cin>>v[i];
    }
}

template<class T>
void vector<T>::modify()   //Function for Modifying the Co-ordinates
{
    int elem;
      cout<<"\nEnter index of element you want to modify ";
    cin>>elem;

    if(elem>size)
    {
    cout<<"Index position cann't be greater than size of vecotr";
    }
    else
    {
    if(elem<0)
    {
          cout<<"Index can not be negative. So initializing as 0";
        elem=0;
    }
    cout<<"\nEnter new value:\n";
    cin>>v[elem];
    }
}
```

```
template<class T>
void vector<T>::display()  //Displaying the vector
{
        cout<<"\nVector : (";
        for(int i=0; i<size; i++)
        {
            cout<<v[i];
            if(i!=size-1)
                cout<<",";
        }
        cout<<")";
}

int main()
{
    vector <int>v;
    v.display();
    v.modify();
    v.display();
    return 0;
}

Output:
Enter Number of Co-ordinates : 3

Enter 3 Co-ordinates :
 1
 7
 2

Vector : (1,7,2)
Enter index of element you want to modify 1

Enter new value:
100

Vector : (1,100,2)
```

9 Write a C++ program to find largest number from given two numbers of different data types using a function template. (S-18)

```
#include <iostream>
using namespace std;
template <class T>

T Large(T a, T b)
{
    return a>b?a:b  ;
}

int main()
{
    int i1, i2;
    float f1, f2;
    char c1, c2;

    cout << "Enter two integers:\n";
    cin >> i1 >> i2;
    cout << Large(i1, i2) <<" is larger." << endl;

    cout << "\nEnter two floating-point numbers:\n";
    cin >> f1 >> f2;
    cout << Large(f1, f2) <<" is larger." << endl;

    cout << "\nEnter two characters:\n";
    cin >> c1 >> c2;
    cout << Large(c1, c2) << " has larger ASCII value.";
}
```

```
Output:
Enter two integers:
5 2
5 is larger.

Enter two floating-point numbers:
4.9 7.1
7.1 is larger.

Enter two characters:
b h
h has larger ASCII value.
```

10 Describe the syntax and use of set () function with suitable example. (W-18)

```
#include<iostream>
#include<set>
using namespace std;

int main()
{
    set<string> setOfNum;

    // Lets insert four elements
    setOfNum.insert("first");
    setOfNum.insert("second");
    setOfNum.insert("third");
    setOfNum.insert("first"); //first will not insert again

    // Only 3 elements will be inserted
    cout<<"Set Size = "<<setOfNum.size()<<endl;

    // Iterate through all the elements in a set and display values.
    set<string>::iterator it;
    for(it=setOfNum.begin(); it!=setOfNum.end(); ++it)
        cout << ' ' << *it;
    cout<<"\n";
    return 0;
}

Output:
Set Size = 3
 first second third
```