# Institute of Space Technology Islamabad

## Assignment 02,03

## Submitted to:

Ma’am Tayyiba

## Submitted by:

Muhammad Bin Nasir

220201012

# Question 01

In this task, we implement multiple inheritance. We create two classes, a ‘Mammal’ and a ‘Bird’ class. Then we create another class which inherits both the classes, we name the third class ‘Organism’. Consider including the following attributes and functionalities to your classes.

**classMammal**

- voidsetMammalName(string);
- voidshowMammal();

**classBird**

- voidsetBirdName(string);
- voidshowBird();

**classOrganism**

- voidsetOrganismName(string);
- voidshowOrganism();
- char* getOrganismName();

## Code:

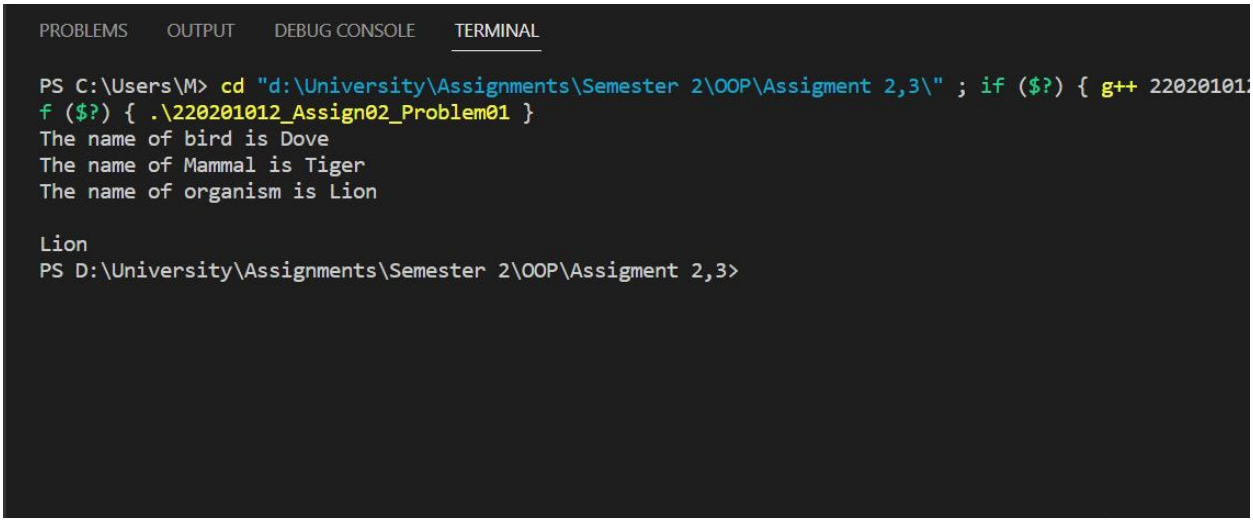
```
#include <iostream>
#include <string>
using namespace std;

class Mammal
{
    string MammalName;

    public:
    Mammal()
    {
        MammalName = "Null";
    }
    string getMammalName()
    {
        return MammalName;
    }
    void setMammalName(string Name)
    {
        MammalName = Name;
    }
    void showMammal()
    {
        cout<<"The name of Mammal is "<<MammalName<<endl;
    }
};

```

```
class Bird
{
    string BirdName;

    public:
    Bird()
    {
        BirdName = "Null";
    }
    string getBirName()
    {
        return BirdName;
    }
    void setBirdName(string Name)
    {
        BirdName = Name;
    }
    void showBird()
    {
        cout<<"The name of bird is "<<BirdName<<endl;
    }
};
```

```
50
51   class Organism:public Bird,public Mammal
52   {
53       string OrganismName;
54
55       public:
56       Organism()
57       {
58           OrganismName = "Null";
59       }
60
61       void setOrganismName(string name)
62       {
63           OrganismName = name;
64       }
65       void showOrganism()
66       {
67           cout<<"The name of organism is "<<OrganismName<<endl;
68       }
69       char* getOrganismName()
70       {
71           char* name;
72           name = &OrganismName[0];
73           return name;
74       }
75   };
```

```
77   int main()
78   {
79       Organism org;
80       org.setBirdName("Dove");
81       org.setMammalName("Tiger");
82       org.setOrganismName("Lion");
83       org.showBird();
84       org.showMammal();
85       org.showOrganism();
86       cout<<endl<<org.getOrganismName();
87       return 0;
88   }
```

# Output:




# Question 02

In this task, we would try to find out areas of various shapes using inheritance. You need to create a class name 'Base'. Define two protected data members of float type for length and width –'LengthorBase' and 'WidthorHeight'. Now you need to define constructor(s), setter(s) and a getter for the base class. Next, you need to define three child classes, 'Rectangle', 'Square' and 'Triangle'. Every class must contain a constructor that invokes the parent class constructor whenever a child instance is created. Following the same approach, create the particular getter, setter(s) and area functions for each of the class. Area of a class must return a value accordingly. Use a suitable inheritance.

**classBase**

- Base(float, float){}
- voidbaseGetter() const{}

**classRectangle**

- Rectangle(float, float);
- voidrectangleGetter(){}
- voidrectangleSetter(){}

- voidrectangleSetter(float, float){}
- floatareaOfRectangle(){}

**classSquare**

- Square(float, float);
- voidsquareGetter(){}
- voidsquareSetter(){}
- voidsquareSetter(float, float){}
- floatareaOfSquare(){}

**classTriangle**

- Triangle(float, float);
- voidtriangleGetter(){}
- voidtriangleSetter(){}
- voidtriangleSetter(float, float){}
- floatareaOfTriangle(){}

## Code:

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

class Base
{
    protected:
    float Length_or_Base;
    float Width_or_Height;

    public:
    Base()
    {
        cout<<"called"<<endl;
        Length_or_Base = 0.0;
        Width_or_Height = 0.0;
    }
    Base(float base,float height)
    {
        Length_or_Base = base;
        Width_or_Height = height;
    }
    void setLength(float base)
    {
        Length_or_Base = base;
    }
    void setWidth(float height)
    {
        Width_or_Height = height;
    }
    float getLength()
    {
        return Length_or_Base;
    }
    float getWidth()
    {
        return Width_or_Height;
    }
};

class Rectangle:public Base
{
    public:
    Rectangle()
    {
        Length_or_Base = 0.0;
        Width_or_Height = 0.0;
    }
    Rectangle(float base,float height)
    {
        Length_or_Base = base;
        Width_or_Height = height;
    }
    void rectangleSetter(float base,float height)
    {
        Length_or_Base = base;
        Width_or_Height = height;
    }
```

```
58      void rectangleGetter()
59      {
60          cout<<"Length is "<<Length_or_Base;
61          cout<<"Width is "<<Width_or_Height;
62      }
63      float areaOfRectangle()
64      {
65          return Length_or_Base * Width_or_Height;
66      }
67  };
68
69  class Triangle:public Base
70  {
71      public:
72      Triangle()
73      {
74          Length_or_Base = 0.0;
75          Width_or_Height = 0.0;
76      }
77      Triangle(float base,float height)
78      {
79          Length_or_Base = base;
80          Width_or_Height = height;
81      }
82      void triangleSetter(float base,float height)
83      {
84          Length_or_Base = base;
85          Width_or_Height = height;
86      }
```

```
    void triangleGetter()
    {
        cout<<"Length is "<<Length_or_Base;
        cout<<"Width is "<<Width_or_Height;
    }
    float areaOfTriangle()
    {
        return Length_or_Base * Width_or_Height /2;
    }
};

class Square:public Base
{
    public:
    Square()
    {
        Length_or_Base = 0.0;
        Width_or_Height = 0.0;
    }
    Square(float base,float height)
    {
        Length_or_Base = base;
        Width_or_Height = height;
    }
    void squareSetter(float base,float height)
    {
        Length_or_Base = base;
        Width_or_Height = height;
    }
    void squareGetter()
    {
        cout<<"Length is "<<Length_or_Base;
        cout<<"Width is "<<Width_or_Height;
    }
    float areaOfSquare()
    {
        return Length_or_Base * Width_or_Height;
    }
};

int main()
{
    Square sqr;
    Triangle trgl;
    Rectangle rct;

    sqr.setLength(4);
    sqr.setWidth(4);
    cout<<endl<<"The length of square = "<<sqr.getLength()<<endl;
    cout<<"Area of square = "<<sqr.areaOfSquare()<<endl<<endl;

    trgl.setLength(3);
    trgl.setWidth(5);
    cout<<"Base of triangle = "<<trgl.getWidth()<<endl;
    cout<<"Height of triangle = "<<trgl.getLength()<<endl;
    cout<<"Area of triangle = "<<trgl.areaOfTriangle()<<endl<<endl;

    rct.setLength(3);
    rct.setWidth(5);
    cout<<"Length of rectangle = "<<rct.getLength()<<endl;
    cout<<"Width of rectangle = "<<rct.getWidth()<<endl;
    cout<<"Area of rectangle = "<<rct.areaOfRectangle()<<endl;

    return 0;
}
```

## Output:

```
PS C:\Users\M> cd "d:\University\Assignments\Semester 2\OOP\Assigment 2,3\" ; if ($?) { g++ 22020
 { .\220201012_Assign_02_03_Q2 }
called
called
called

The length of square = 4
Area of square = 16

Base of triangle = 5
Height of triangle = 3
Area of triangle = 7.5

Length of rectangle = 3
Width of rectangle = 5
Area of rectangle = 15
PS D:\University\Assignments\Semester 2\OOP\Assigment 2,3>
```

# Question 03

We want to calculate the total marks of each student of a class in Physics, Chemistry and Mathematics and the average marks of the class. The number of students in the class are entered by the user. Create a class named Marks with data members for roll number, name and marks.Create three other classes inheriting the Marks class, namely Physics, Chemistry and Mathematics, which are used to define marks in individual subject of each student. Roll number of each student will be generated automatically.

## Code:

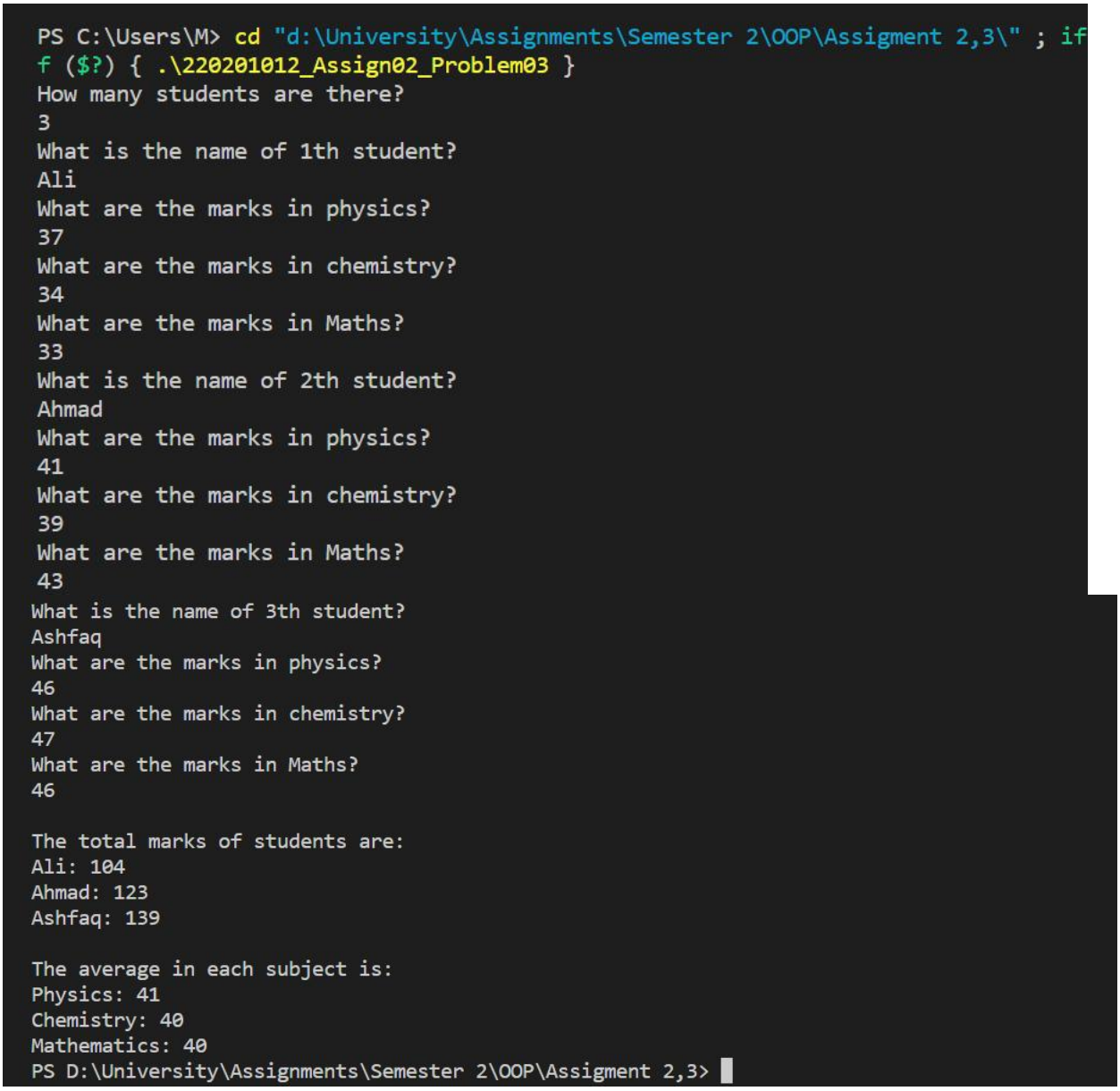
```cpp
#include <iostream>
using namespace std;

class Marks
{
    protected:
    int rollNo;
    string name;
    float marks;

    public:
    Marks()
    {
        rollNo = RollGenerate();
        name = "NULL";
        marks = 0;
    }
    private:
    int RollGenerate()
    {
        static int temp0 = 0;
        return ++temp0;
    }

};

class Physics:public Marks
{
    public:
    Physics()
    {}
    Physics(string Name,float mark)
    {
        name = Name;
        marks = mark;
    }
    void setName(string Name)
    {
        name = Name;
    }
    void setMarks(float mark)
    {
        marks = mark;
    }
    string getName()
    {
        return name;
    }
    int getRoll()
    {
        return rollNo;
    }
    float getMarks()
    {
        return marks;
    }
};
```

```
class Mathematics:public Marks
{
    public:
    Mathematics()
    {}
    Mathematics(string Name,float mark)
    {
        name = Name;
        marks = mark;
    }
    void setName(string Name)
    {
        name = Name;
    }
    void setMarks(float mark)
    {
        marks = mark;
    }
    string getName()
    {
        return name;
    }
    int getRoll()
    {
        return rollNo;
    }
    float getMarks()
    {
        return marks;
    }
};
```

```
class Chemistry:public Marks
{
    public:
    Chemistry()
    {}
    Chemistry(string Name,float mark)
    {
        name = Name;
        marks = mark;
    }
    void setName(string Name)
    {
        name = Name;
    }
    void setMarks(float mark)
    {
        marks = mark;
    }
    string getName()
    {
        return name;
    }
    int getRoll()
    {
        return rollNo;
    }
    float getMarks()
    {
        return marks;
    }
};
```

```cpp
int main()
{
    int temp,mathMarks,phyMarks,chemMarks,temp1=0,temp2=0,temp3=0,temp4=0,tempo;
    string name;
    cout<<"How many students are there?"<<endl;
    cin>>temp;

    Physics* phy = new Physics[temp];
    Chemistry* chem = new Chemistry[temp];
    Mathematics* math = new Mathematics[temp];

    for (int i=0;i<temp;i++)
    {
        cout<<"What is the name of "<<i+1<<"th student?"<<endl;
        cin>>name;
        cout<<"What are the marks in physics?"<<endl;
        cin>>phyMarks;
        cout<<"What are the marks in chemistry?"<<endl;
        cin>>chemMarks;
        cout<<"What are the marks in Maths?"<<endl;
        cin>>mathMarks;
        phy[i] = Physics(name,phyMarks);
        chem[i] = Chemistry(name,chemMarks);
        math[i] = Mathematics(name,mathMarks);
    }

    cout<<endl;
    cout<<"The total marks of students are: "<<endl;
    for (int i=0;i<temp;i++)
    {
        temp1 += phy[i].getMarks();
        temp2 += chem[i].getMarks();
        temp3 += math[i].getMarks();
        tempo = phy[i].getMarks() + chem[i].getMarks() + math[i].getMarks();
        cout<<phy[i].getName()<<": "<<tempo<<endl;
    }


    cout<<endl;
    cout<<"The average in each subject is: "<<endl;
    cout<<"Physics: "<<temp1/temp<<endl;
    cout<<"Chemistry: "<<temp2/temp<<endl;
    cout<<"Mathematics: "<<temp3/temp<<endl;

    return 0;
}
```

# Output:




# Question 04

Consider a class Computer having Two fields (i.e. companyName, price) and A single function named show() A class named Desktop inherits Computer class and adds fields representing color, monitor size, and processor type and Override function named show() to display values of its all attributes A class named Laptop inherits Computer class and adds fields representing color, size, weight, and processor type and Override function named show() to display values of its all

attributes In Main() instantiate objects of derived classes to access respective show() functions to see the polymorphic behavior

# Code:

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

class Computer
{
    string companyName;
    int price;

    public:
    Computer()
    {
        companyName = "NULL";
        price = 0;
    }
    Computer(string name,int Price)
    {
        companyName = name;
        price = Price;
    }
    void setCompany(string name)
    {
        companyName = name;
    }
    void setPrice(int pri)
    {
        price = pri;
    }
    string getCompany()
    {
        return companyName;
    }
    int getPrice()
    {
        return price;
    }
    void show()
    {
        cout<<"Company: "<<companyName<<endl;
        cout<<"Price: "<<price<<endl;
    }
};

class Desktop:public Computer
{
    string color;
    float monitorSize;
    string processorType;

    public:
    Desktop()
    {
        color = "NULL";
        monitorSize = 0;
        processorType = "NULL";
    }
    Desktop(string name,int Price,string Color,int size,string type):Computer(name,Price),
    color(Color),monitorSize(size),processorType(type){}
    void setColor(string Color)
    {
```

```
        color = Color;
    }
    void setType(string type)
    {
        processorType = type;
    }
    void setSize(int size)
    {
        monitorSize = size;
    }
    string getColor()
    {
        return color;
    }
    int getSize()
    {
        return monitorSize;
    }
    string getType()
    {
        return processorType;
    }

    void show()
    {
        Computer::show();
        cout<<"Color: "<<color<<endl;
        cout<<"Monitor Size: "<<monitorSize<<endl;
        cout<<"Processor Type: "<<processorType<<endl;
    }
};
class Laptop:public Computer
{
    string color;
    float size;
    float weight;
    string processorType;

    public:
    Laptop()
    {
        color = "NULL";
        size = 0;
        weight = 0;
        processorType = "NULL";
    }
    Laptop(string name,int Price,string Color,float Size,float Weight,string type):Computer(name,Price),
    color(Color),size(Size),processorType(type),weight(Weight){}
    void setColor(string Color)
    {
        color = Color;
    }
    void setType(string type)
    {
        processorType = type;
    }
    void setSize(float Size)
    {
        size = Size;
    }
```

```
    void setWeight(float Weight)
    {
        weight = Weight;
    }
    string getColor()
    {
        return color;
    }
    int getSize()
    {
        return size;
    }
    string getType()
    {
        return processorType;
    }

    void show()
    {
        Computer::show();
        cout<<"Color: "<<color<<endl;
        cout<<"Size: "<<size<<endl;
        cout<<"Weight: "<<weight<<endl;
        cout<<"Processor Type: "<<processorType<<endl;
    }
};

int main()
{
    Laptop lap("Dell",150000,"Black",17,160,"Ryzen 7 5700U");
    Desktop dsk("Alienware",275000,"White",32,"Ryzen 7 5800X3D");
    lap.show();
    cout<<endl;
    dsk.show();
    cout<<endl;

    return 0;
}
```

# Output:

```
PS C:\Users\M> cd "d:\University\Assignments\Semester 2\OOP\Assigment 2,3\" ; if ($?) { g++ 2
 { .\220201012_Assign_02_03_Q4 }
Company: Dell
Price: 150000
Color: Black
Size: 17
Weight: 160
Processor Type: Ryzen 7 5700U

Company: Alienware
Price: 275000
Color: White
Monitor Size: 32
Processor Type: Ryzen 7 5800X3D

PS D:\University\Assignments\Semester 2\OOP\Assigment 2,3>
```

# Question 05

Write a class Distance that holds distances or measurements expressed in feets and inches. This class has two private data members:

●feet: An integer that holds the feet.

●inches: An integer that holds the inches.

●Write a constructor with default parameters that initializes each data member of the class. If inches are greater than equal to 12 then they must be appropriately converted to corresponding feet.

●Generate appropriate getter-setter functions for the data members.

1. void setFeet(int f) and int getFeet()const
2. void setInches(int i) It should ensure proper conversion to feet.
3. int getInches() const

●Define an operator '+' that overloads the standard '+' math operator and allows one Distance object to be added to another. Distance operator+ (const Distance &obj).

●Define an operator -function that overloads the standard '-' math operator and allows subtracting one Distance object from another. Distance operator-(const Distance &obj)

●Define an operator= function that overloads the = operator and assign one Distance object to another. const Distance operator=(const Distance &obj)

## Code:

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

class Distance
{
    int feet,inches;

    public:
    Distance()
    {
        feet = 0;
        inches = 0;
    }
    Distance(int a, int b)
    {
        while (b>=12)
        {
            b %= 12;
            a += 1;
        }
        feet = a;
        inches = b;
    }
    void setFeet(int f)
    {
        feet = f;
    }
    int getFeet()
    {
        return feet;
    }
    void setInches(int i)
    {
        while (i>=12)
        {
            i %= 12;
            feet += 1;
        }
        inches = i;
    }
    int getInches()
    {
        return inches;
    }

    Distance operator + (const Distance &obj)
    {
        Distance d;
        d.feet = feet + obj.feet;
        d.inches = inches + obj.inches;
        while (d.inches>=12)
        {
            d.inches %= 12;
            d.feet += 1;
        }
        return d;
    }

```

```
    Distance operator - (const Distance &obj)
    {
        Distance d;
        if (inches<obj.inches)
        {
            d.inches = obj.inches - inches;
            d.feet = feet - obj.feet -1;
        }
        else
        {
            d.inches = inches - obj.inches;
            d.feet = feet - obj.feet;
        }
        return d;
    }

    Distance operator = (const Distance &obj)
    {

        feet = obj.feet;
        inches = obj.inches;

    return *this;

    }
};

int main()
{
    Distance d1(6,11);
    cout<<"Distance 1 is:"<<endl<<d1.getFeet()<<" feet and "
        <<d1.getInches()<<" inches."<<endl<<endl;

    Distance d2(5,13);
    cout<<"Distance 2 is:"<<endl<<d2.getFeet()<<" feet and "
        <<d2.getInches()<<" inches."<<endl<<endl;

    Distance d3;
    d3.setFeet(7);
    d3.setInches(14);
    cout<<"Distance 3 is:"<<endl<<d3.getFeet()<<" feet and "
        <<d3.getInches()<<" inches."<<endl<<endl;


    Distance d4;
    d4 = d2 + d3;
    cout<<"Distance 4 = Distance 2 + Distance 3:"<<endl<<d4.getFeet()<<" feet and "
        <<d4.getInches()<<" inches."<<endl<<endl;

    Distance d5;
    d5 = d1 - d2;
    cout<<"Distance 5 = Distance 1 - Distance 2:"<<endl<<d5.getFeet()<<" feet and "
        <<d5.getInches()<<" inches."<<endl<<endl;


    Distance d6;
    d6 = d1;
    cout<<"Distance 6 = Distance 1 (assignment operator):"<<endl<<d6.getFeet()<<" feet and "
        <<d6.getInches()<<" inches."<<endl<<endl;


    return 0;

}
```

## Output:

```
PS C:\Users\M> cd "d:\University\Assignments\Semester 2\OOP\Assigment 2,3\" ; if ($?) { g++ 2
 { .\220201012_Assign_02_03_Q5 }
Distance 1 is:
6 feet and 11 inches.

Distance 2 is:
6 feet and 1 inches.

Distance 3 is:
8 feet and 2 inches.

Distance 4 = Distance 2 + Distance 3:
14 feet and 3 inches.

Distance 5 = Distance 1 - Distance 2:
0 feet and 10 inches.

Distance 6 = Distance 1 (assignment operator):
6 feet and 11 inches.

PS D:\University\Assignments\Semester 2\OOP\Assigment 2,3>
```

# Question 06

You are supposed to construct a class named Matrix that shall contains private data member:

- matrix:integer type array of size 3 by 3;

Define a constructor that should have default parameters that can set all elements to 0. Moreover, define a function called setMatrixValues(int matrixArray[3][3]) that can assign values of matrixArray (provided to the function in argument) to the matrix array (data member of the class Matrix) of the class. Along with it, define a function called displayMatrix()that can display all values of the matrix.Moreover, you are instructed to construct a program that can perform operator overloading for the following operators + (plus), -(minus), and == (equal).

# Code:

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

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

    public:
    Matrix()
    {
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                matrix[i][j] = 0;
            }
        }
    }

    void setMatrixValues(int matrixArray[3][3])
    {
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                matrix[i][j] = matrixArray[i][j];
            }
        }
    }

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

    Matrix operator + (Matrix &M)
    {
        Matrix temp;
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                temp.matrix[i][j] = matrix[i][j] + M.matrix[i][j];
            }
        }
        return temp;
    }
```

```
    Matrix operator - (Matrix &M)
    {
        Matrix temp;
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                temp.matrix[i][j] = matrix[i][j] - M.matrix[i][j];
            }
        }
        return temp;
    }

    bool operator == (Matrix &M)
    {
        int count = 0;
        for (int i=0;i<3;i++)
        {
            for (int j=0;j<3;j++)
            {
                if (matrix[i][j] != M.matrix[i][j])
                {
                    count += 1;
                }
            }
        }
        if (count==0)
        {
            return true;
        }
        else
        {
            return false;
        }

    }
};

int main()
{
    Matrix m1,m2,m3,m4;
    bool temp,temp1;

    int mat[3][3];
    for (int i=0;i<3;i++)
    {
        for (int j=0;j<3;j++)
        {
            mat[i][j] = i*j;
        }
    }

    m1.setMatrixValues(mat);
    m2.setMatrixValues(mat);
    cout<<"Matrix 1 is:"<<endl;
    m1.display();
    cout<<"Matrix 2 is:"<<endl;
    m2.display();
    m3 = m1 + m2;
    cout<<"Matrix 3 = Matrix 1 + Matrix 2:"<<endl;
    m3.display();
```

```
    m4 = m1 - m2;
    cout<<"Matrix 4 = Matrix 1 - Matrix 2:"<<endl;
    m4.display();
    temp = m1==m2;
    temp1 = m1==m3;
    if (m1==m2)
    {
        cout<<"Matrix 1 and Matrix 2 are equal."<<endl;
    }
    else
    {
        cout<<"Matrix 1 and Matrix 2 are not equal."<<endl;
    }
    if (m1==m3)
    {
        cout<<"Matrix 1 and Matrix 3 are equal."<<endl;
    }
    else
    {
        cout<<"Matrix 1 and Matrix 3 are not equal."<<endl;
    }

    return 0;
}
```

# Output:

```
PS C:\Users\M> cd "d:\University\Assignments\Semester 2\OOP\Assigment 2,3\" ; if ($?) { g++ 2202
 { .\220201012_Assign_02_03_Q6 }
Matrix 1 is:
[ 0 0 0 ]
[ 0 1 2 ]
[ 0 2 4 ]


Matrix 2 is:
[ 0 0 0 ]
[ 0 1 2 ]
[ 0 2 4 ]


Matrix 3 = Matrix 1 + Matrix 2:
[ 0 0 0 ]
[ 0 2 4 ]
[ 0 4 8 ]


Matrix 4 = Matrix 1 - Matrix 2:
[ 0 0 0 ]
[ 0 0 0 ]
[ 0 0 0 ]


Matrix 1 and Matrix 2 are equal.
Matrix 1 and Matrix 3 are not equal.
PS D:\University\Assignments\Semester 2\OOP\Assigment 2,3>
```

# Question 07

We want to create a class of Product that contains multiple private data members such as

- quantity: An integer that holds a count value.

- ●objCount: A static integer that holds that count of objects.
- ●serialNo: An integer that holds the serial number of objects of a specific product (assume the single object of Product class).
- ●Define a constructor that can accept two arguments i.e., totalQuantityOfProduct and serialNumberOfProduct; and assign it to the respective data members of the class. Moreover, the static member be shall be initialized from the outside of the class with zero by using scope resolution operator.
- ●Define operator = that add the value of quantity to the left hand operand. i.e. c2=c1 (the quantity of object c2 shall be incremented with the quantity of c1).
- ●Define unary operator -that inverts the value of quantity for product class and should allow the statements like c1 -= 4;

## Code:

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

class Product
{
    int quantity;
    static int objCount;
    int serialNo;

    public:
    Product()
    {
        quantity = 0;
        serialNo = 0;
    }
    Product(int totalQuantityOfProduct,int serialNumberOfProduct)
    {
        quantity = totalQuantityOfProduct;
        serialNo = serialNumberOfProduct;
    }

    void setQuantity(int quan)
    {
        quantity = quan;
    }
    void setSerial(int serial)
    {
        serialNo = serial;
    }
    int getQuantity()
    {
        return quantity;
    }
    int getSerial()
    {
        return serialNo;
    }

    Product operator =(Product &obj)
    {
        quantity += obj.quantity;
        return *this;
    }
    Product operator -= (Product &obj)
    {
        quantity -= obj.quantity;
        return *this;
    }
    Product operator -= (int num)
    {
        quantity -= num;
        return *this;
    }

};

```

```cpp
int main()
{
    Product prd1(10,15),prd2,prd3,prd4;
    prd2.setQuantity(20);
    prd2.setSerial(101);

    cout<<"Product 1:"<<endl;
    cout<<"Quantity: "<<prd1.getQuantity()<<endl;
    cout<<"Serial no: "<<prd1.getSerial()<<endl<<endl;

    cout<<"Product 2:"<<endl;
    cout<<"Quantity: "<<prd2.getQuantity()<<endl;
    cout<<"Serial no: "<<prd2.getSerial()<<endl<<endl;

    prd2 = prd1;

    cout<<"Product 2 after adding values of Product 1:"<<endl;
    cout<<"Quantity: "<<prd2.getQuantity()<<endl;
    cout<<"Serial No: "<<prd2.getSerial()<<endl<<endl;

    prd2 -= prd1;

    cout<<"Product 2 after subtracting values of Product 1:"<<endl;
    cout<<"Quantity: "<<prd2.getQuantity()<<endl;
    cout<<"Serial No: "<<prd2.getSerial()<<endl<<endl;

    prd2 -= 4;

    cout<<"PRroduct 2 after subtracting 4 from the quantity:"<<endl;
    cout<<"Quantity: "<<prd2.getQuantity()<<endl;
    cout<<"Serial No: "<<prd2.getSerial()<<endl<<endl;

    return 0;
}
```

# Output:

```
PS C:\Users\M> cd "d:\University\Assignments\Semester 2\OOP\Assigment 2,3\" ; if ($?) { g++ 2
 { .\220201012_Assign_02_03_Q7 }
Product 1:
Quantity: 10
Serial no: 15

Product 2:
Quantity: 20
Serial no: 101

Product 2 after adding values of Product 1:
Quantity: 30
Serial No: 101

Product 2 after subtracting values of Product 1:
Quantity: 20
Serial No: 101

PRroduct 2 after subtracting 4 from the quantity:
Quantity: 16
Serial No: 101

PS D:\University\Assignments\Semester 2\OOP\Assigment 2,3>
```