

**HANDOUTS | MATERIAL
HIGHLIGHTED AND PROVIDED BY :**

VU TALK

CS304p Complete Preparation In One Video:

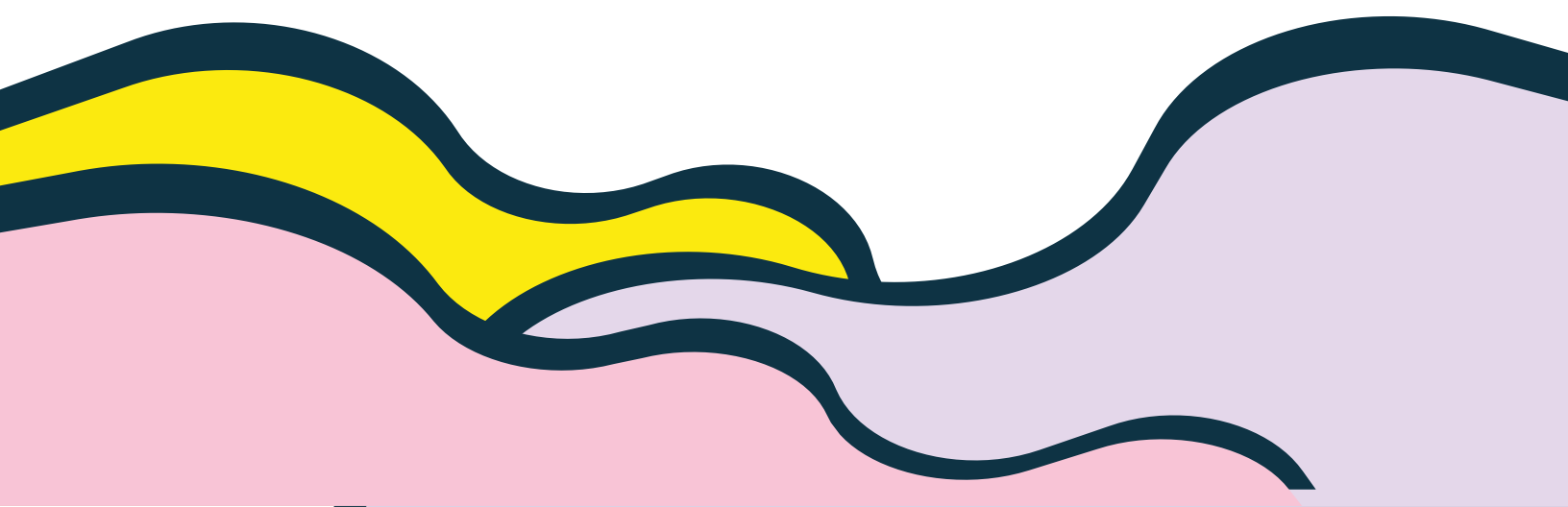
<https://youtu.be/smWe0kYpuBI>

CLICK TO JOIN OUR WHATSAPP GROUP : [click me](#)

VU TALK YouTube CHANNEL : [click me](#)

For Any Help Contact Me On Whatsapp : [click me](#)

Follow VU TALK on Whatsapp Channel : [click me](#)



Lab 01 Problem Statement

Consider the following scenario and identify all **objects**, their **attributes** and **behaviors**. In Virtual University **bookshop**, the **handouts** of all **courses** are provided at reasonable price. The **handouts** are made in accordance with the **video lectures** recorded by famous **professors**. It is a step to facilitate **students** to easily digest the **course contents**.

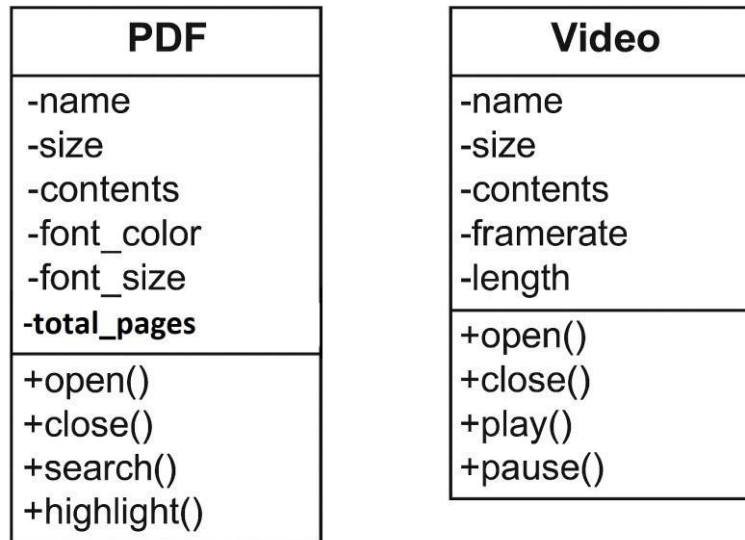
Solution :

Objects	Attributes (characteristics)	Behaviors (operations)
Bookshop	Name Address (Location) Phone Number List of books	Set Order Get Order Confirm Order Deliver Order Update Inventory Payment receive/collect
Handout	Title No. of topics/chapters Page Number Author Publisher Creation Date Modify date Edition	Upload Download Edit Print Read Write Share View
Course	Course code Course name Course instructor Credit hour Total marks Grading scheme No. of lectures	Add information View Information Edit Information Read Write
Video Lectures	Video number Name Duration Video quality Type Video URL	View Record Upload Download Stop Share Pause Forward Reverse Increase speed Decrease speed Increase volume Decrease volume Playback

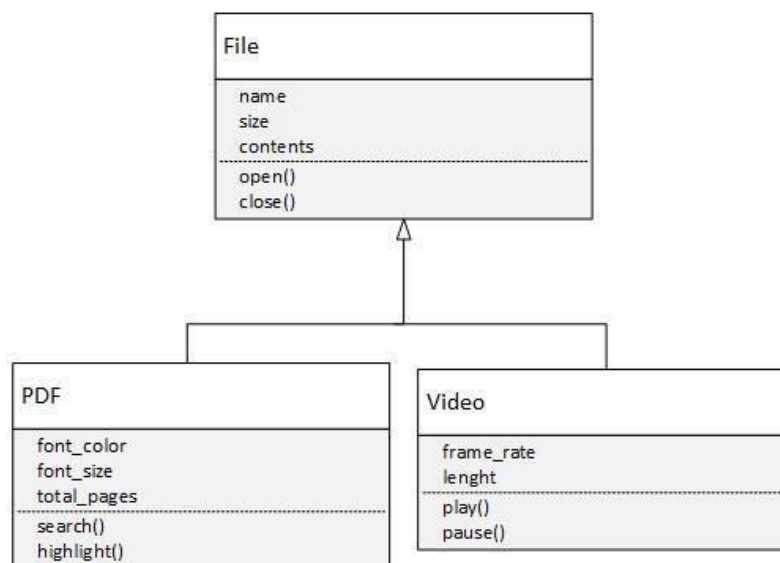
Professor	Employee ID Name Age Gender Qualification Specialization Designation Experience Salary No. of taught subjects	Teach Develop assignment Solve student's problems Develop quiz Develop GDB Prepare Exam Take Exam Invigilate Mark papers Mark activities Give grades Prepare result
Student	Student ID Name Age Gender Student program Semester CGPA No. of courses	Study Give exam Attempt activities Taking lectures Write notes Making time table
Course content	No. of topics Names Topic number Topic Description Composed by Publish by Written by	Edit View Share Write Download

Lab 02 Problem Statement

In the below diagram, we have two different types of file classes (PDF and Video). You are required to apply the concept of generalization to the given classes and draw a class diagram showing this generalization relationship.



Solution :



Lab 03 Problem Statement

Write a program which consists of a class named **Room** having two data members **Height** and **Width**, the class should also consist of **three constructors** i.e., Default constructor, one argument constructor and two arguments constructor.

Sample Output:

After running your program, the following screen shot should display.



```
C:\Users\amna.bibi\Desktop\C++ Codes\Untitled1.exe
Default Constructor is Called
Height: 0
Width: 0

One Argument Constructor is Called
Height: 10
Width: 0

Two Arguments Constructor is Called
Height: 10
Width: 20

Press any key to continue . . . _
```

Solution :

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

class Room {
private:
    double height;
    double width;
public:
    Room()
    {
        height = 0;
        width = 0;
        cout << "Default constructor is called.." << endl;
        cout << "Height: " << height << endl;
        cout << "Width: " << width << endl<<endl;
    }

    Room(double h)
    {
        height = h;
        width = 0;
        cout << "One argument constructor is called.." << endl;
        cout << "Height: " << height << endl;
        cout << "Width: " << width << endl << endl;
    }

    Room(double h, double w)
    {
        height = h;
        width = w;
        cout << "Two argument constructor is called.." << endl;
        cout << "Height: " << height << endl;
        cout << "Width: " << width << endl << endl;
    }
};

int main()
{
    Room obj1;
    Room obj2(10);
    Room obj3(10, 20);
}
```

Lab 04 Problem Statement

Write a C++ coding program that creates a class Pyramid. The class should contain a static data member “Pcount” to store the total number of Pyramid’s objects.

In the main() function, dynamically create three objects of the Pyramid class and print the value of “Pcount”. Afterward, delete any two objects of Pyramid and print the value of “Pcount” again.

Sample Output

Following is a sample output for the above scenario:

A screenshot of a Windows command prompt window. The title bar shows the file path "E:\Programming Labs\Lab 4. Static Data Member\Lab 4.exe". The window has standard Windows controls (minimize, maximize, close). The command prompt area is black with white text. The output of the program is as follows:

```
Pyramid Constructor Called..  
Pyramid Constructor Called..  
Pyramid Constructor Called..  
Total Objects : 3  
  
Pyramid Destructor Called..  
Pyramid Destructor Called..  
Total Objects : 1  
Press any key to continue . . .
```

Solution :

```
#include<iostream>  
using namespace std;  
class pyramid {  
public:  
    static int pcount;  
    pyramid()  
    {  
        cout << "Pyramid Constructor Called.." << endl;  
        pcount++;  
    }  
    ~pyramid()  
    {  
        cout << "Pyramid Destructor Called.." << endl;  
        pcount--;  
    }  
  
    static int get_pcount()  
    {  
        return pcount;  
    }  
};  
int pyramid::pcount = 0;  
int main()  
{  
    pyramid* p1 = new pyramid;
```

```

    pyramid* p2 = new pyramid;
    pyramid* p3 = new pyramid;

    cout << "Total Objects : " << pyramid::get_pcount() << endl;
    cout << "\n";
    delete p1;
    delete p2;
    cout << "Total Objects : " << pyramid::get_pcount();
}

```

Another solution of the same lab with separated interface & implementation

```

#include <iostream>
using namespace std;

class pyramid {
public:
    static int pcount;

    pyramid();           // Constructor
    ~pyramid();          // Destructor
    static int get_pcount(); // Static function
};

// Initialize static variable
int pyramid::pcount = 0;

// Static function definition
int pyramid::get_pcount()
{
    return pcount;
}

// Constructor definition
pyramid::pyramid()
{
    cout << "Pyramid Constructor Called.." << endl;
    pcount++;
}

// Destructor definition
pyramid::~~pyramid()
{
    cout << "Pyramid Destructor Called.." << endl;
    pcount--;
}

int main()
{
    pyramid* p1 = new pyramid;
    pyramid* p2 = new pyramid;
    pyramid* p3 = new pyramid;
}

```

```

    cout << "Total Objects : " << pyramid::get_pcount() << endl << endl;

    delete p1;
    delete p2;

    cout << "Total Objects : " << pyramid::get_pcount() << endl;

    return 0;
}

```

Lab 05 Problem Statement

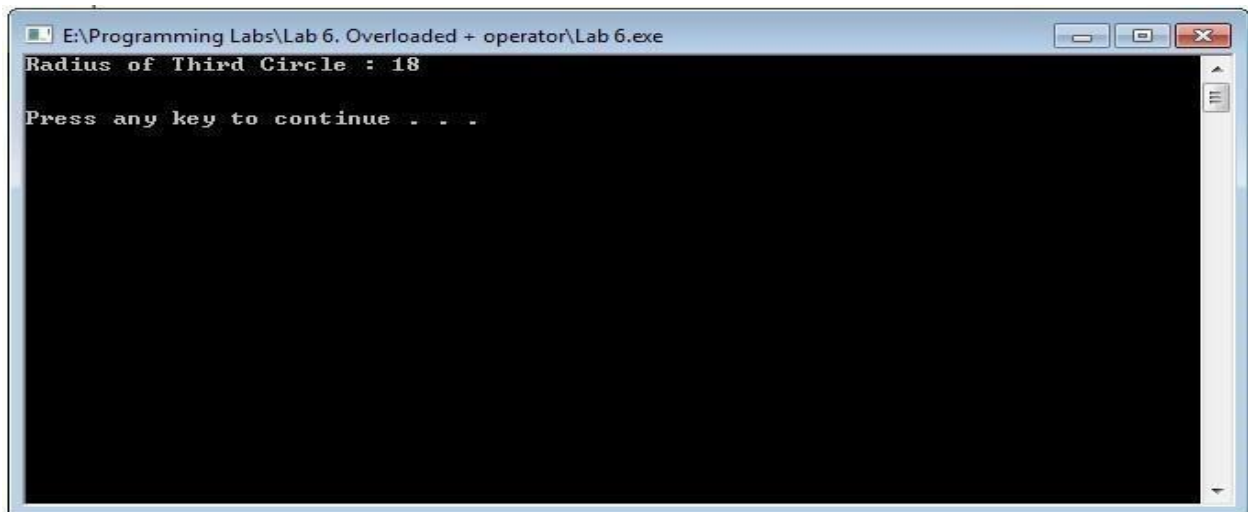
Write a C++ program that consists of a class named **Circle** which has one data member **radius** and **setter** and **getter functions**. Overload the + **operator** for this class.

In the main () function, create three objects c1, c2, and c3. Call setter function for objects c1 and c2. The program should be able to execute the following statement; c3=c1+c2;

Call the getter function for c3.

After running your program, the following screen should display.

Sample Output :



Solution :

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

class Circle {
private:
    double radius;
public:
    void set_data(double radius)
    {
        this->radius = radius;
    }

    double get_data()
    {
        return radius;
    }

    // ----- OPERATOR OVERLOADING PART -----
    // Yeh function + operator ko overload karta hai.
    // Jab bhi hum likhenge: c3 = c1 + c2;
    // Toh compiler is function ko call karega.
    // c1 hamesha "this" pointer se represent hoga,
    // aur jo object hum parameter me bhejte hain (yaha c2), wo "c" ke andar aayega.
    Circle operator +(Circle c)
    {
        Circle temp; // ek naya Circle object banaya jo result store karega.

        // this->radius = current object ka radius (e.g., c1)
        // c.radius      = jo object argument me aaya (e.g., c2)
        // dono radii ko add karke temp ke radius me daal diya
        temp.radius = this->radius + c.radius;

        // ab temp object jisme result hai, use return kar diya
        return temp;
    }
};

int main()
{
    Circle c1;
    Circle c2;
    Circle c3;

    c1.set_data(10);
    c2.set_data(20);

    // Yeh line call karegi operator+ function ko
    c3 = c1 + c2;

    cout << "Radius of third Circle: " << c3.get_data();
}
```

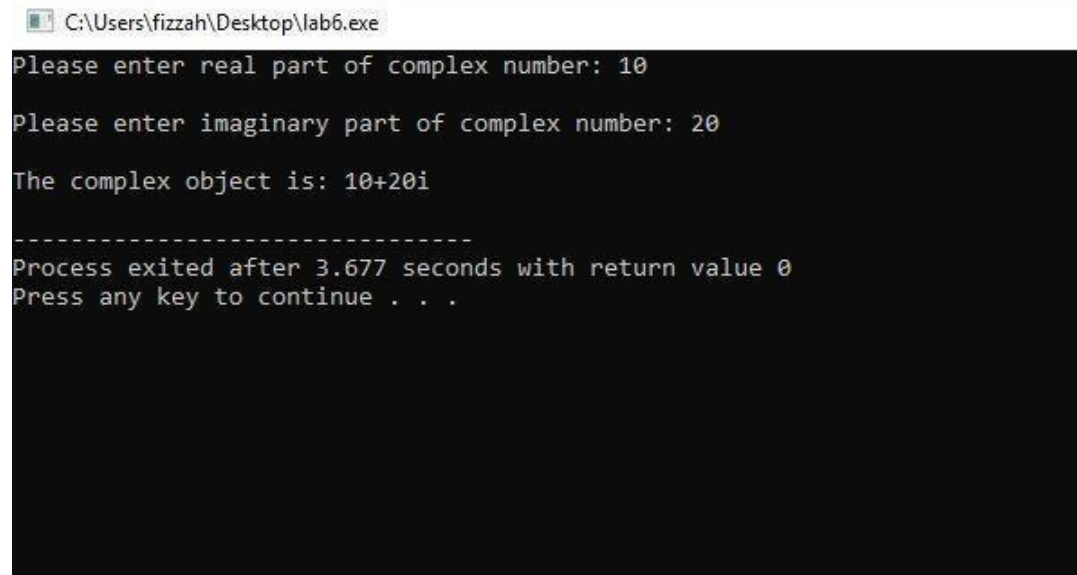
Lab 06 Covering Lectures: Lectures 19-20 Problem Statement

Write a C++ program that consists of a class named **Complex** which has data members **real** and **imaginary**.

Overload the **stream insertion operator (<<)** and **stream extraction operator (>>)** for this class.

In the main () function, create an object com1 and take input using the stream extraction operator and print output using the stream insertion operator

After running your program, the following screen should display.



```
C:\Users\fizzah\Desktop\lab6.exe
Please enter real part of complex number: 10
Please enter imaginary part of complex number: 20
The complex object is: 10+20i
-----
Process exited after 3.677 seconds with return value 0
Press any key to continue . . .
```

Solution :

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

class Complex {
public:
    int real;
    int imaginary;
};

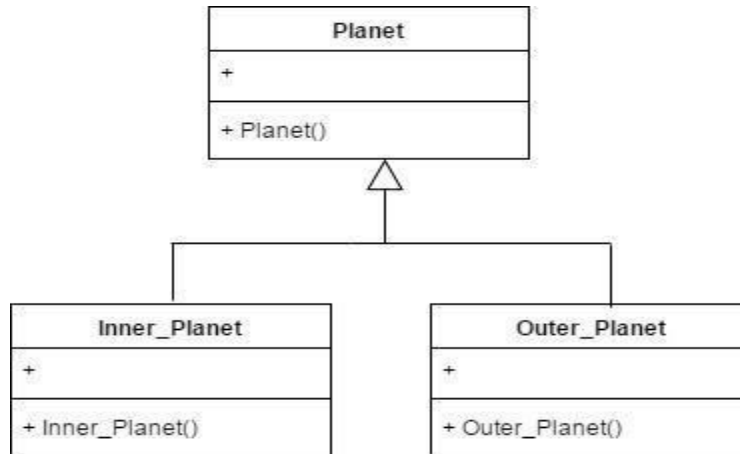
istream& operator >>(istream& input, Complex& c)
{
    cout << "Please enter real part of complex number: ";
    input >> c.real;
    cout << "Please enter imaginary part of complex number: ";
    input >> c.imaginary;
    return input;
}

ostream& operator <<(ostream& output, Complex& c)
{
    output << "\n The complex object is: " << c.real << "+" << c.imaginary <<
    "i";
    return output;
}

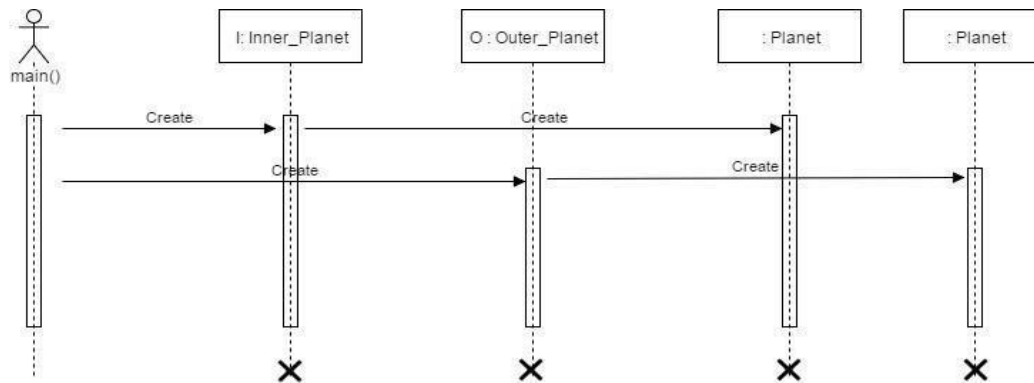
int main()
{
    Complex obj;
    cin >> obj;
    cout << obj;
}
```

Lab 07 Covering Lecture: Lecture 22 Problem Statement

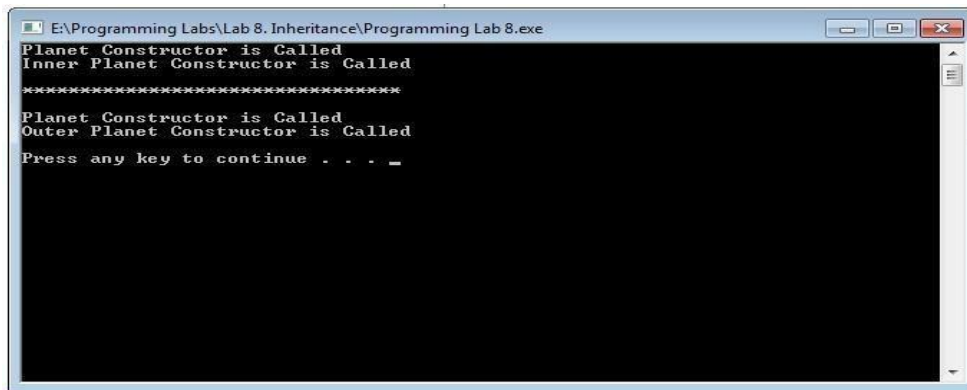
Write a C++ program that consists of a class named **Planet** which has no data member and only contains a default constructor. Your program should contain two more classes **Inner_Planet** and **Outer_Planet**. Both of these classes should also contain a default constructor. Your program should implement the concept of inheritance between “**Planet class, Inner_Planet class, and Outer_Planet class**”.



Sequence Diagram



Sample Output :



```
E:\Programming Labs\Lab 8. Inheritance\Programming Lab 8.exe
Planet Constructor is Called
Inner Planet Constructor is Called
*****
Planet Constructor is Called
Outer Planet Constructor is Called
Press any key to continue . . . _
```

Solution :

```
#include<iostream>
using namespace std;
class Planet {
public:
    Planet()
    {
        cout << "Planet Constructor is Called" << endl;
    }
};

class Inner_Planet :public Planet {
public:
    Inner_Planet()
    {
        cout << "Inner Planet Constructor is Called" << endl;
    }
};

class Outer_Planet :public Planet {
public:
    Outer_Planet()
    {
        cout << "Outer Planet Constructor is Called" << endl;
    }
};

int main()
{
    Inner_Planet obj1;
    cout << "\n*****\n\n";
    Outer_Planet obj2;
}
```

Lab 08 Problem Statement

Write a C++ program that should consist of the following four classes.

1. Base
2. Derived_Private
3. Derived_Protected
4. Derived_Public

“Base” class should consist of following data members along with following type

Data member	Access Specifier
secret	private
protected	protected
access	public

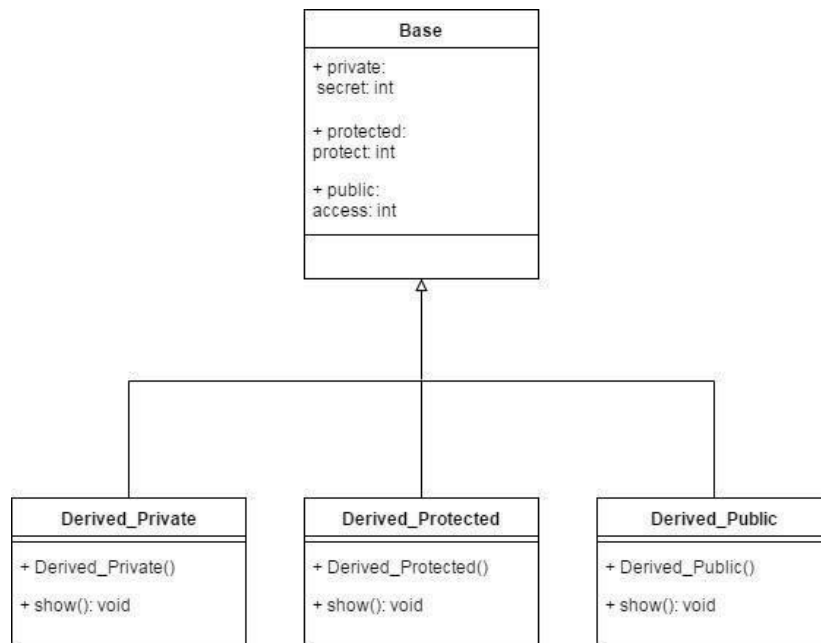
Further, “Base” class should contain default constructor to initialize its data members.

Other three classes should be inherited from “Base” class with respect to following type of inheritance.

Class	Inheritance type
Derived_Private	Private
Derived_Protected	Protected
Derived_Public	Public

All three derived classes should contain **show() function**, which should display the values of **secret**, **protect** and **access** data member of base class. On running this program, your compiler will generate compile time errors. Carefully observe for which type of base class data member, compiler is generating error.

Class Diagram :



Compilation Results :

Compiler (9)				Resources	Compile Log	Debug	Find Results	Close
Line	Col	File	Message					
		E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	In member function 'void Derived_Private::show()':					
7	7	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] 'int Base::secret' is private					
33	25	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] within this context					
		E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	In member function 'void Derived_Protected::show()':					
7	7	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] 'int Base::secret' is private					
44	25	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] within this context					
		E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	In member function 'void Derived_Public::show()':					
7	7	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] 'int Base::secret' is private					
55	25	E:\Programming Labs\Lab 9. Inheritance 2\Lab 9.cpp	[Error] within this context					

Solution :

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

class Base {
private:
    int secret;
protected:
    int protect;
public:
    int access;

    Base()
    {
        secret = 10;
        protect = 20;
        access = 30;
    }
};

class Derived_Private :private Base{
public:
    void show()
    {
        cout << "secret: " << secret << endl;
        cout << "protect: " << protect<<endl;
        cout << "access: " << access<<endl;
    }
};

class Derived_Protected :protected Base {
public:
    void show()
    {
        cout << "secret: " << secret << endl;
        cout << "protect: " << protect << endl;
        cout << "access: " << access << endl;
    }
};

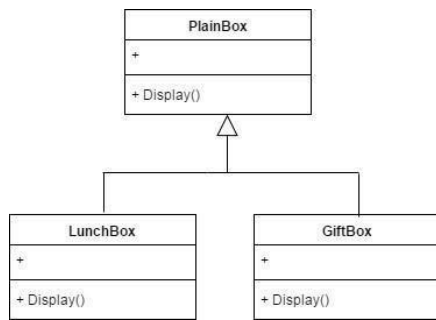
class Derived_Public :public Base {
public:
    void show()
    {
        cout << "secret: " << secret << endl;
        cout << "protect: " << protect << endl;
        cout << "access: " << access << endl;
    }
};

int main()
{
    Derived_Private obj1;
    obj1.show();

    Derived_Protected obj2;
    obj2.show();
}
```

```
    Derived_Public obj3;  
    obj3.show();  
}
```

Lab 09 Lectures Covered: 28-30 Problem Statement Class Diagram



Part (1):

Write C++ program which should contain three classes as follows:

1. PlainBox
2. LunchBox
3. GiftBox

“LunchBox” and “GiftBox” classes should be publically inherited from PlainBox Class.

All three classes should contain Display() function, which should display respective class name.

In main() function, create objects of LunchBox and GiftBox classes dynamically. Static type of these objects should be “PlainBox”. Using object of PlainBox, call Display() function.

Following is a sample output for the above scenario:

```
Select C:\Users\fizzah\Desktop\Untitled1.exe
Plain Box Class
Plain Box Class
Press any key to continue . . .
```

Solution :

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

class PlainBox {
public:
    void display()
    {
        cout << "Plain Box Class" << endl;
    }
};

class LunchBox :public PlainBox {
public:
    void display()
    {
        cout << "Lunch Box Class" << endl;
    }
};

class GiftBox :public PlainBox{
public:
    void display()
    {
        cout << "Plain Box Class" << endl;
    }
};

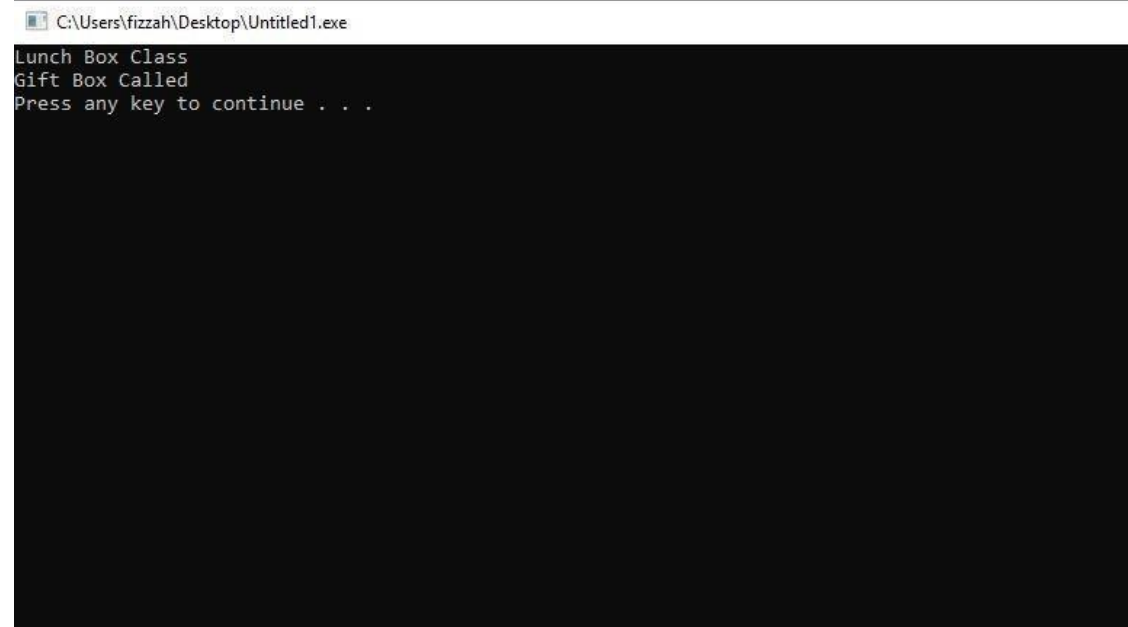
int main()
{
    PlainBox* ptr1 = new LunchBox;
    ptr1->display();

    PlainBox* ptr2 = new GiftBox;
    ptr2->display();
}
```

Part (2):

Make Display() function of “PlainBox” class virtual and then run the program. Your output should be as follows:

Sample Output



```
C:\Users\fizzah\Desktop\Untitled1.exe
Lunch Box Class
Gift Box Called
Press any key to continue . . .
```

Solution :

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

class PlainBox {
public:
    virtual void display()
    {
        cout << "Plain Box Class" << endl;
    }
};

class LunchBox :public PlainBox {
public:
    void display()
    {
        cout << "Lunch Box Class" << endl;
    }
};

class GiftBox:public PlainBox{
public:
    void display()
    {
        cout << "Plain Box Class" << endl;
    }
};

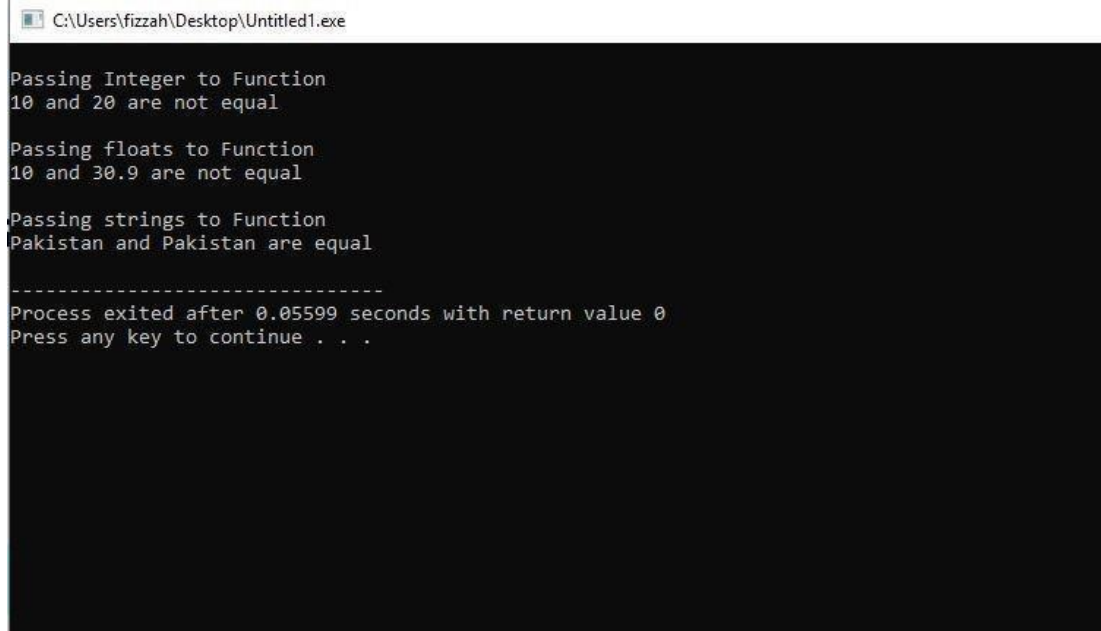
int main()
{
    PlainBox* ptr1 = new LunchBox;
    ptr1->display();

    PlainBox* ptr2 = new GiftBox;
    ptr2->display();
}
```

Lab 10 Problem Statement

Write C++ code for a class Temp. Class has no data members but a function Compare. The function accepts two values either integer, double, characters or strings and compare them. The function returns true if the two passed values are equal otherwise returns false.

Sample Output



```
C:\Users\fizzah\Desktop\Untitled1.exe
Passing Integer to Function
10 and 20 are not equal

Passing floats to Function
10 and 30.9 are not equal

Passing strings to Function
Pakistan and Pakistan are equal

-----
Process exited after 0.05599 seconds with return value 0
Press any key to continue . . .
```

Solution :

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

class temp {
public:
    template<class T>
    bool compare(T x, T y)
    {
        return (x == y);
    }
};

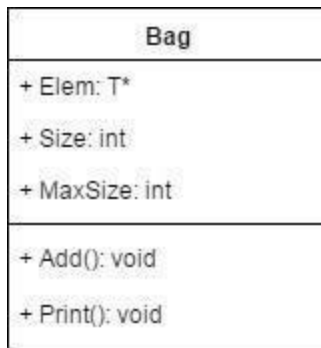
int main()
{
    temp obj;
    cout << "Passing Integer to Function" << endl;
    if (obj.compare(10,20))
    {
        cout << "10 and 20 are equal" << endl;
    }
    else
    {
        cout << "10 and 20 are not equal" << endl;
    }

    cout << "\nPassing floats to Function" << endl;
    if (obj.compare(10.0, 30.9))
    {
        cout << "10 and 30.9 are equal" << endl;
    }
    else
    {
        cout << "10 and 30.9 are not equal" << endl;
    }

    cout << "\nPassing strings to Function" << endl;
    if (obj.compare("Pakistan","Pakistan"))
    {
        cout << "Pakistan and Pakistan are equal" << endl;
    }
    else
    {
        cout << "Pakistan and Pakistan are not equal" << endl;
    }
}
```

Lab 11 Problem Statement

Write C++ code for the following template Bag class.

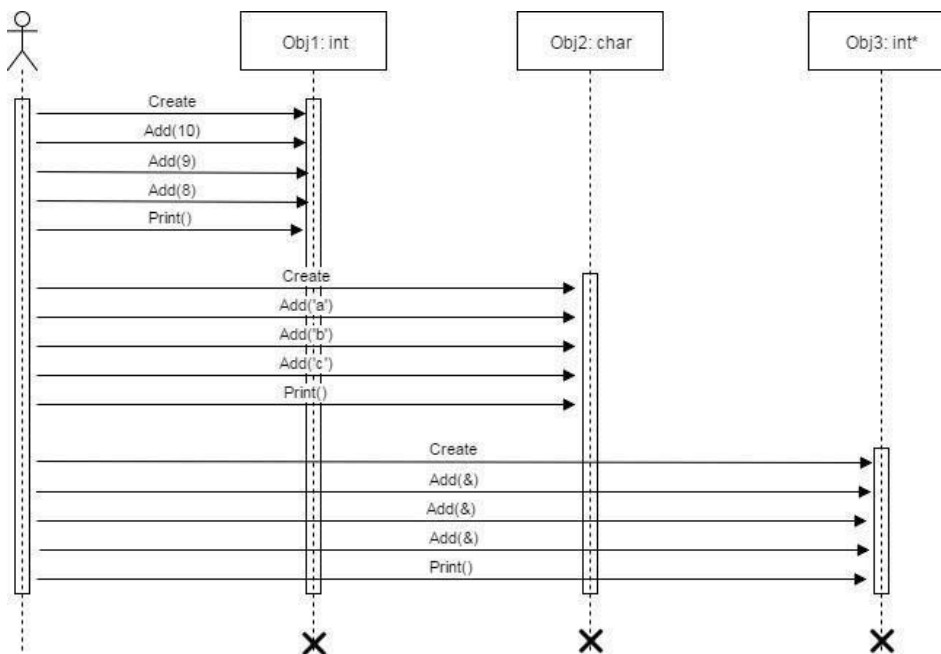


Element is an array that stores the elements present in the bag. Elements can store different types of elements. Size is the number of elements present in Bag and MaxSize hold the max number of element's a bag can have.

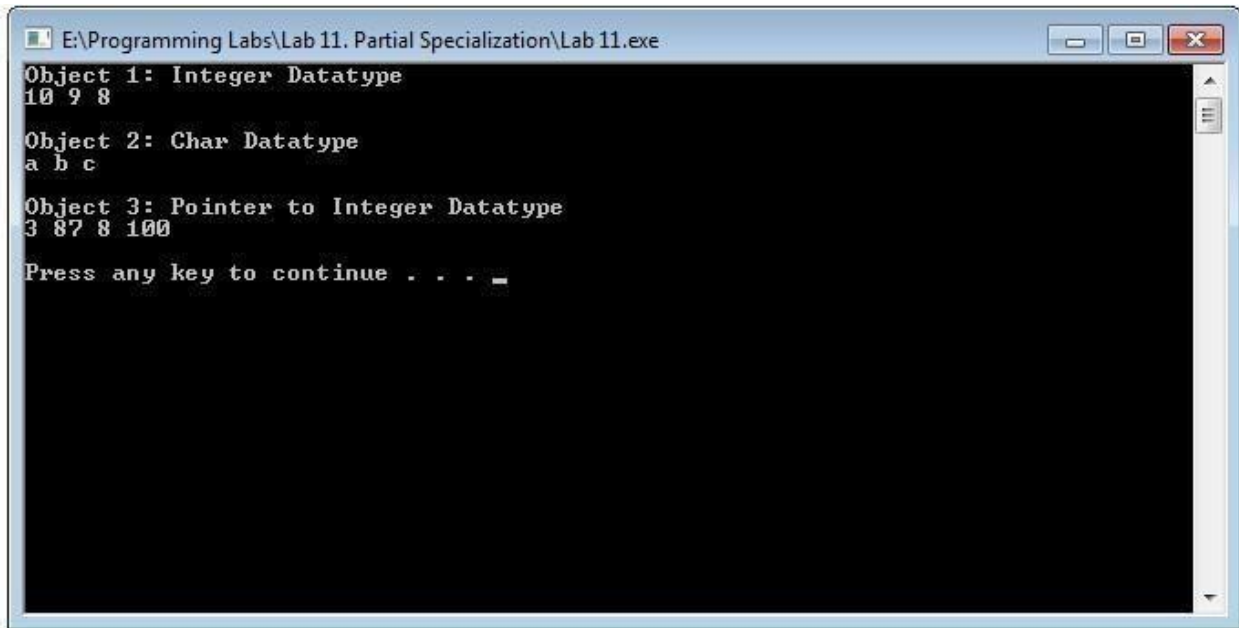
Add method is adding elements present in the bag into the array Elem and print will print the added elements on screen. Implement partial specialization for the given class to handle pointers to any datatype.

In main create 3 objects one for each integer, character, and integer type pointer datatypes. Add at least three elements in each.

Sequence Diagram :



Sample Output :



```
E:\Programming Labs\Lab 11. Partial Specialization\Lab 11.exe
Object 1: Integer Datatype
10 9 8
Object 2: Char Datatype
a b c
Object 3: Pointer to Integer Datatype
3 87 8 100
Press any key to continue . . . _
```

Solution :

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

template<class T>

class Bag
{
private:
    T* array;
    int size, maxSize;

public:
    Bag()
    {
        array = new T[3];
        size = 0;
        maxSize = 0;
    }

    void Add(T value)
    {
        array[size++] = value;
        maxSize *= 2;
    }

    void Print()
    {
        for (int i = 0; i < size; i++)
        {
            cout << array[i] << " ";
        }
    }
};

template<class T>

class Bag<T*>
{
private:
    T* array;
    int size, maxSize;

public:
    Bag()
    {
        array = new T[3];
        size = 0;
        maxSize = 0;
    }

    void Add(T* value)
    {
        array[size++] = *value;
        maxSize *= 2;
    }
}
```

```

void Print()
{
    for (int i = 0; i < size; i++)
    {
        cout << array[i] << " ";
    }
};

```

```

int main()
{
    Bag<int> IB;
    Bag<char> CB;
    Bag<int*> PB;

    int i = 40, j = 50, k = 60;

    IB.Add(10);
    IB.Add(20);
    IB.Add(30);

    CB.Add('A');
    CB.Add('B');
    CB.Add('C');

    PB.Add(&i);
    PB.Add(&j);
    PB.Add(&k);

    cout << "Obj1 Integer Values: ";
    IB.Print();

    cout << "\n\nObj2 Char Values: ";
    CB.Print();

    cout << "\n\nObj3 Pointer Values: ";
    PB.Print();
}

```

Lab 12 Problem Statement

Write C++ code for the following template readData and Subtract classes.

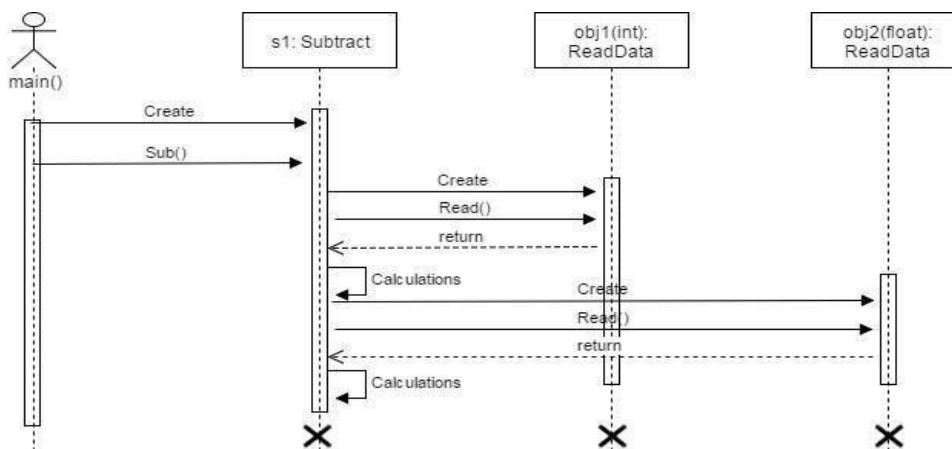


Subtract class is friend of ReadData class and can access the private data members of the Subtract class.

read() method of Subtract class is taking input from user for both data members.

In Subt() function create two objects of ReadData for int and float and perform subtraction on the data for both objects and print the result for each datatype

Sequence Diagram :



Sample Output :

```
E:\Programming Labs\Lab 13. Template&Friend\Lab 13.exe
Integer Type Data
Enter First Number : 67
Enter Second Number : 35
Result of Object 1
Difference = 32
Float Type Data
Enter First Number : 56.789
Enter Second Number : 42.5431
Result of Object 2
Difference = 14.2459
Press any key to continue . . . _
```

Solution :

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

template<class T>
class ReadData{
private:
    T a, b;

public:
    void read()
    {
        cout << "\n\nEnter First Value : ";
        cin >> a;
        cout << "\n\nEnter Second Value : ";
        cin >> b;
    }

    friend class Subtract;
};

class Subtract{
public:
    void Sub()
    {
        ReadData<int> obj1;
        ReadData<float> obj2;

        cout << " Integer Type Data ";
        obj1.read();

        cout << "\n\nObj1 Subtract Result : " << obj1.a - obj1.b;

        cout << "\n\n Float Type Data ";
        obj2.read();

        cout << "\n\nObj2 Subtract Result : " << obj2.a - obj2.b;
    }
};

int main()
{
    Subtract obj;
    obj.Sub();
}
```

Lab 13 Problem Statement

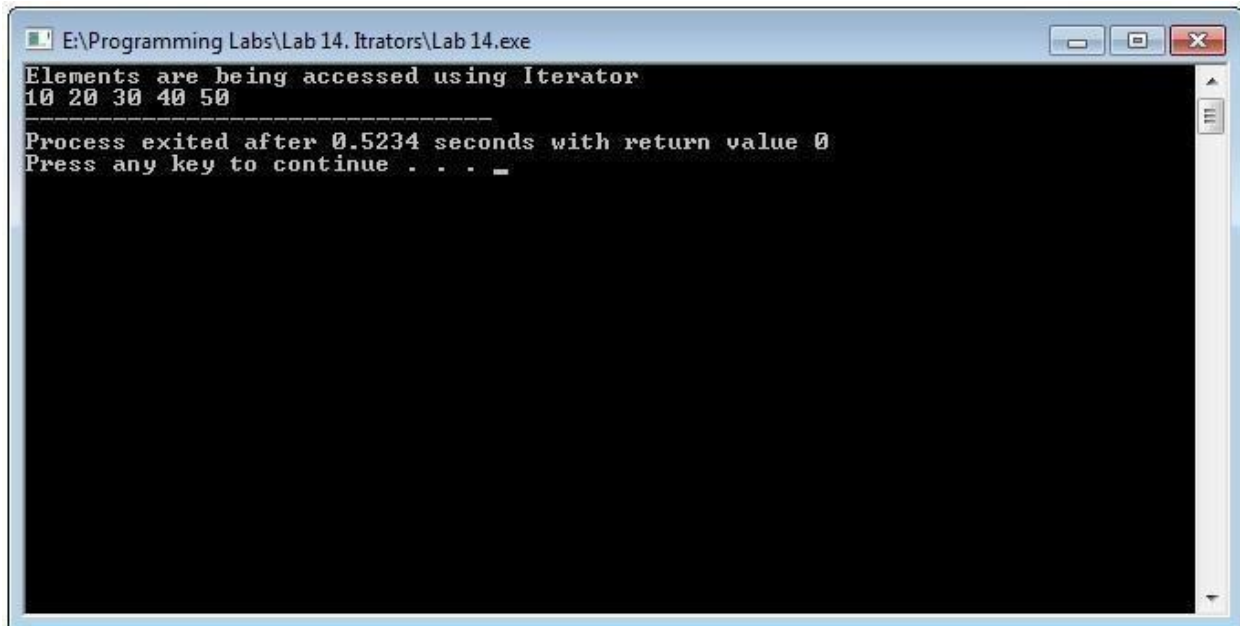
Write a program which should contain an integer Vector to store elements like:

`vector<int> sampleVector;` and an Iterator to access elements like

`vector<int>::iterator sampleIterator;`

The program should store 5 elements 10, 20, 30, 40, 50 using `push_back()` and then show these elements on screen (Iterator will be used to access Vector elements).

Sample Output



```
E:\Programming Labs\Lab 14. Iterators\Lab 14.exe
Elements are being accessed using Iterator
10 20 30 40 50
-----
Process exited after 0.5234 seconds with return value 0
Press any key to continue . . . _
```

Solution :

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

int main()
{
    vector<int> v;
    vector<int>::iterator i;

    v.push_back(10);
    v.push_back(20);
    v.push_back(30);
    v.push_back(40);
    v.push_back(50);

    cout << "Elements are being accessed using iterator"<<endl;
    for (i = v.begin(); i != v.end(); i++)
    {
        cout << *i << " ";
    }

}
```

Lab 14

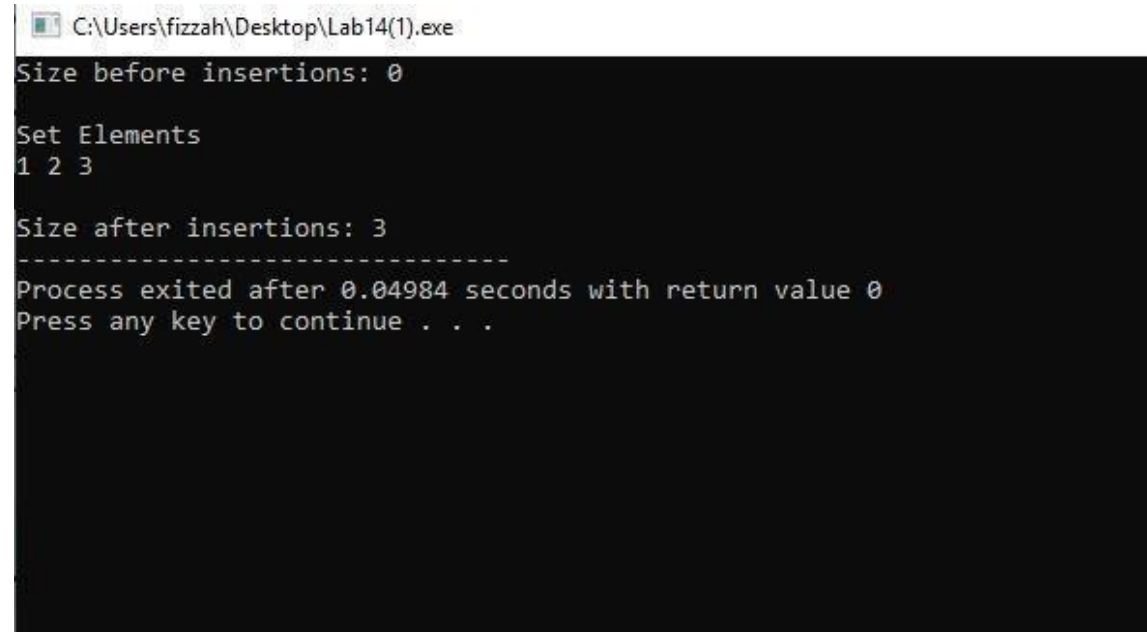
Part 1:

Problem Statement

Write a program that should contain an integer set to store elements and an Iterator to access elements.

The program should store 5 elements 1,2,2,3,3 using insert() and then show the size of set before insertion and after insertion.

Sample Output



```
C:\Users\fizzah\Desktop\Lab14(1).exe
Size before insertions: 0
Set Elements
1 2 3
Size after insertions: 3
-----
Process exited after 0.04984 seconds with return value 0
Press any key to continue . . .
```

Solution :

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

int main()
{
    set<int> s;
    set<int>::iterator i;

    cout << "Before Insertion Size of Set: " << s.size();
    s.insert(1);
    s.insert(2);
    s.insert(2);
    s.insert(3);
    s.insert(3);

    cout << "\n\nSet Elements: "<<endl;
    for (i = s.begin(); i != s.end(); i++)
    {
        cout << *i << " ";
    }

    cout << "\n\nAfter Insertion Size of Set: " << s.size();
}
```

Part 2:

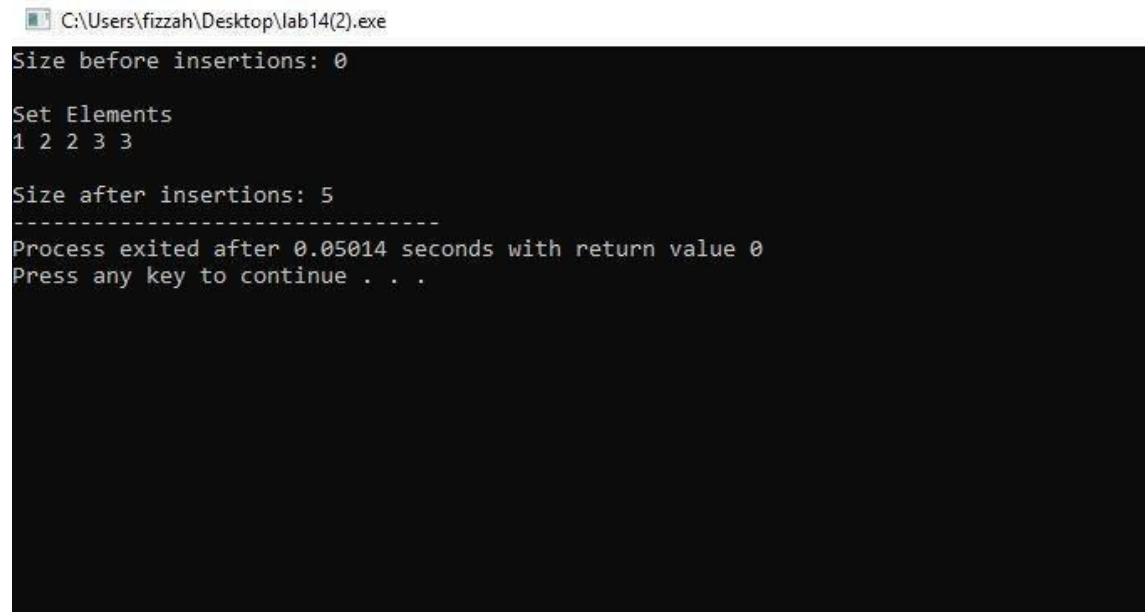
Problem Statement

Write a program that should contain an integer multiset to store elements and an Iterator to access elements.

The program should store 5 elements 1,2,2,3,3 using insert() and then show the size of multiset before insertion and after insertion.

Sample Output

After running your program, the following output screen should display



```
C:\Users\fizzah\Desktop\lab14(2).exe
Size before insertions: 0

Set Elements
1 2 2 3 3

Size after insertions: 5
-----
Process exited after 0.05014 seconds with return value 0
Press any key to continue . . .
```

Solution :

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

int main()
{
    multiset<int> s; // multiset allow duplicates
    multiset<int>::iterator i;

    cout << "Before Insertion Size of Set: " << s.size();
    s.insert(1);
    s.insert(2);
    s.insert(2);
    s.insert(3);
    s.insert(3);

    cout << "\n\nSet Elements: " << endl;
    for (i = s.begin(); i != s.end(); i++)
    {
        cout << *i << " ";
    }

    cout << "\n\nAfter Insertion Size of Set: " << s.size();
}
```

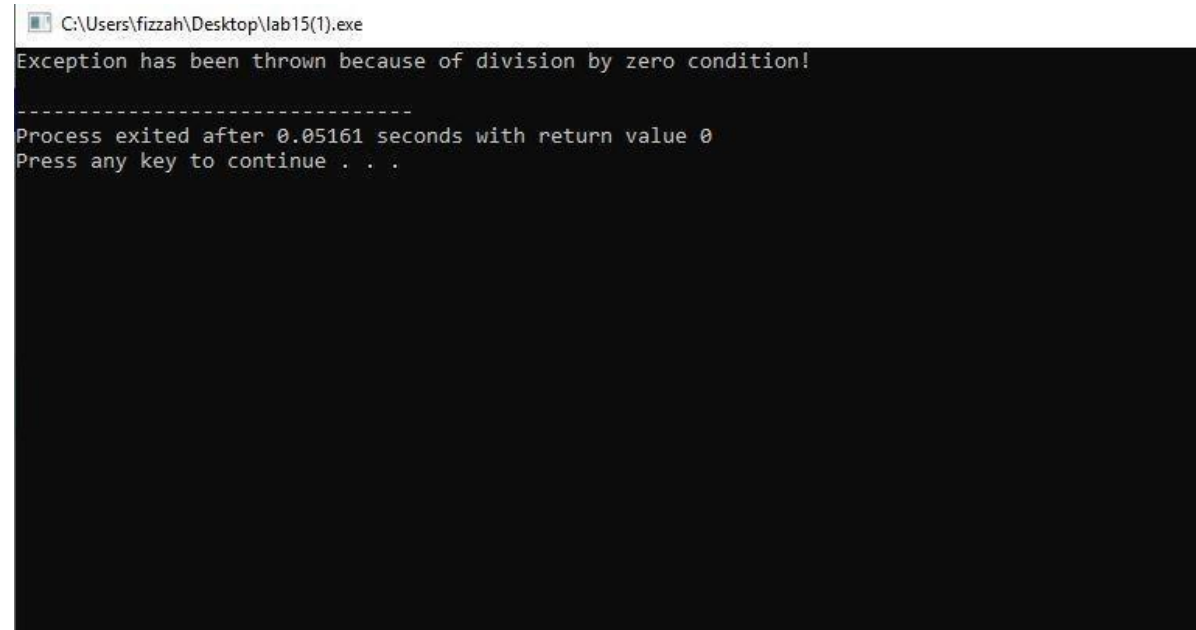
Lab 15

Part 1:

Problem Statement

Write a C++ program to divide the two number and it should throw the exception if number is divided by 0.

Sample Output



```
C:\Users\fizzah\Desktop\lab15(1).exe
Exception has been thrown because of division by zero condition!
-----
Process exited after 0.05161 seconds with return value 0
Press any key to continue . . .
```

Solution :

In case of user defined program you'll be taking the values for x & y from the user.

```
#include<iostream>
using namespace std;
int main()
{
    int x, y;
    cout << "Enter Two Numbers: "<<endl;
    cin >> x >> y;

    try
    {
        if (y == 0)
        {
            throw "Exception has been thrown because of division by zero
condition!";
        }
        else
        {
            cout << "Result: " << (double)x / y;
        }
    }

    catch (const char* msg)
    {
        cout << msg;
    }
}
```

Similarly if you're going to make a predefined program then instead of taking the values from the user, you'll hard code the values for x & y.

```
#include<iostream>
using namespace std;
int main()
{
    int x, y;
    x = 10;
    y = 0;

    try
    {
        if (y == 0)
        {
            throw "Exception has been thrown because of division by zero
condition!";
        }
        else
        {
            cout << "Result: " << (double)x / y;
        }
    }

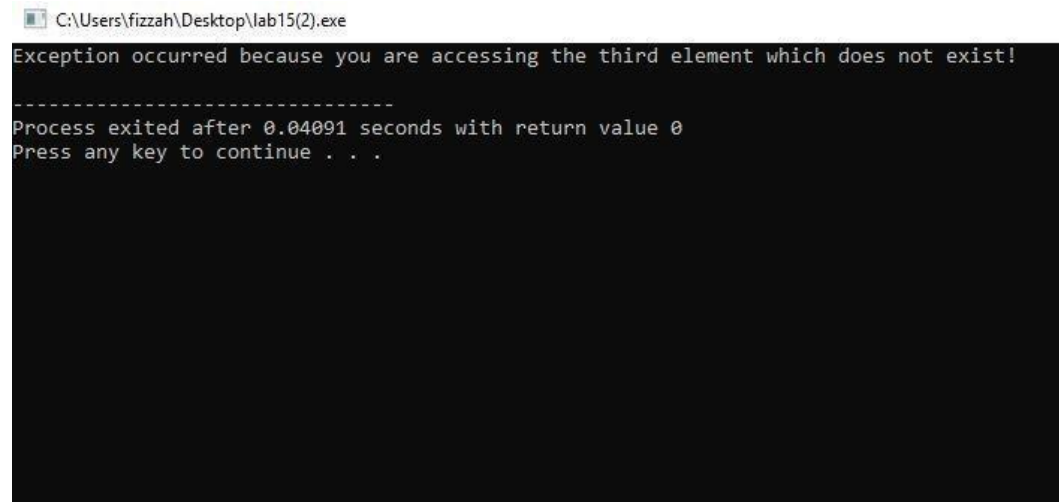
    catch (const char* msg)
    {
        cout << msg;
    }
}
```

Part 2:

Problem Statement

Write a C++ program to create a vector named vec to store integer data. The program should throw exception when try to access the third element of vector vec.

Sample Output



```
C:\Users\fizzah\Desktop\lab15(2).exe
Exception occurred because you are accessing the third element which does not exist!
-----
Process exited after 0.04091 seconds with return value 0
Press any key to continue . . .
```

Solution :

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

int main()
{
    vector<int> vec;

    // Add only two elements (so 3rd element doesn't exist)
    vec.push_back(10);
    vec.push_back(20);

    try
    {
        // Try to access 3rd element (index = 2)
        cout << "Third element: " << vec.at(2) << endl;
    }
    catch (exception& e)
    {
        cout << "Exception occurred because you're accessing the third element which
does not exist!" << endl;
    }
}
```

efforts made by

Broken404 x lost soul