OOP LABORATORY 9

Name: **ANIRBAN HAZRA**
Section: **B-12**
Roll : **2005643**

1. WAP to declare a class which stores a complex number. Include a member function which compares the modulus of the two complex class objects and returns the object with higher value. Include a parameterized constructor which arguments with same name as that of the class data members.

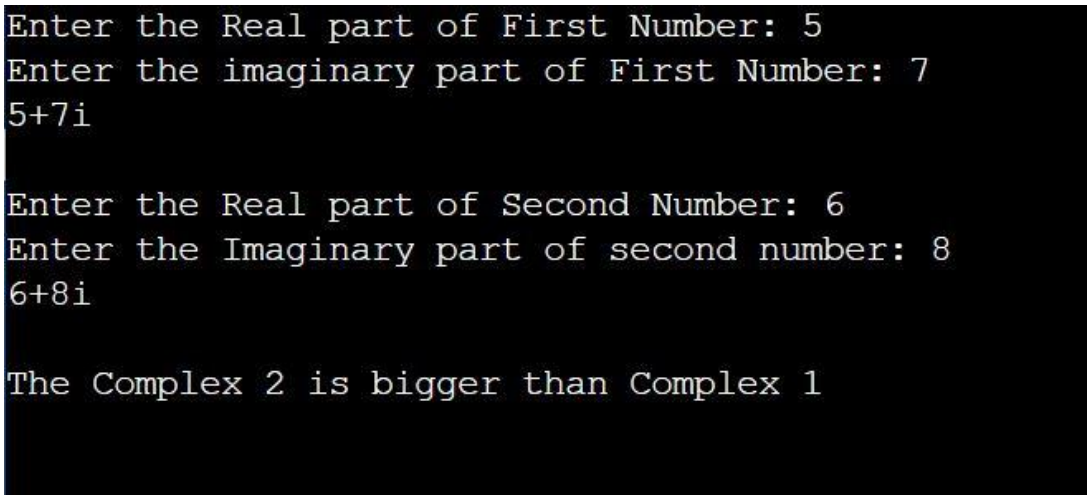
PROGRAM CODE:

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

class Complex
{

private:
  double real, imag;

public:
    Complex ()
  {

    this->real = this->imag = 0;
  }

  Complex (double r, double i)
  {

    this->real = r;
    this->imag = i;
  }

  Complex (Complex & obj)
  {

    this->real = obj.real;
    this->imag = obj.imag;

  }
```

```
int compare (Complex & obj)
{

  double modulusThis =  sqrt (this->real * this->real + this->imag * this->imag);
  double modulusOther = sqrt (obj.real * obj.real + obj.imag * obj.imag);

  if (modulusThis > modulusOther)
    {
        return 1;
    }

  if (modulusThis < modulusOther)
    {
        return -1;
    }

  return 0;
}

void print ()
{

  cout << real << "+" << imag << "i" << endl << endl;
}

double getReal () const
{

  return real;
}

double getImag () const
{

  return imag;
}
void setReal (double re)
{

  real = re;
}
void setImag (double im)
{

  imag = im;
}

};
```

```
int main ()
{
 double real1, imag1, real2, imag2;

 cout << "Enter the Real part of First Number: ";
 cin >> real1;
 cout << "Enter the imaginary part of First Number: ";
 cin >> imag1;
 Complex obj1 (real1, imag1);
 obj1.print ();

 cout << "Enter the Real part of Second Number: ";
 cin >> real2;
 cout << "Enter the Imaginary part of second number: ";
 cin >> imag2;
 Complex obj2 (real2, imag2);
 obj2.print ();

 if (obj1.compare (obj2) == 1)
   {
    cout << "The Complex 1 is bigger than Complex 2\n";
   }
 if (obj1.compare (obj2) == -1)
   {
    cout << "The Complex 2 is bigger than Complex 1\n";
   }
 if (obj1.compare (obj2) == 0)
   {
    cout << "The Complex 2 is equal Complex 1\n";
   }


 return 0;
}
```

OUTPUT :



2. WAP in which there is a global variable, a local variable for main function and a variable in a nested scope inside main, with the same name. Print all the three variables.

PROGRAM CODE:

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

int x= 10;
int main()
{
   int x=20;
   cout<<:: x<<" ";
   cout<<x<<" ";
   {x=30;
    cout<<x<<" ";
   }

   return 0;
}
```

OUTPUT:

```
10 20 30

...Program finished with exit code 0
Press ENTER to exit console.
```

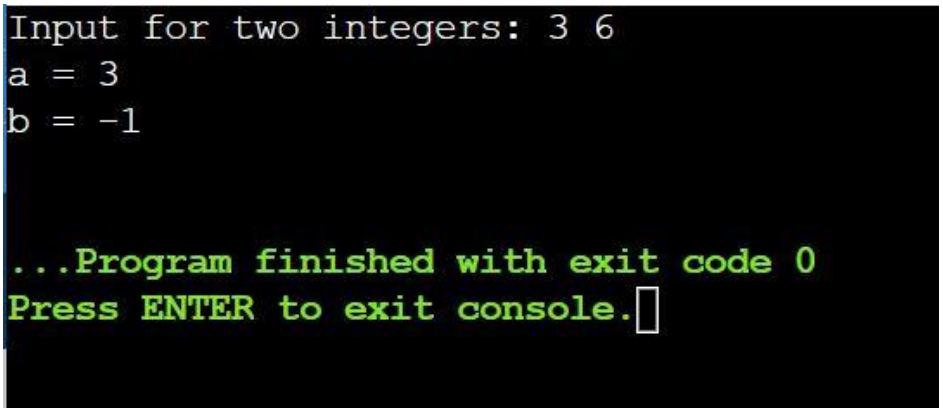
3. WAP to take input for two integer variables. Assign the value -1 to the variable with higher value using a function. [Use return by reference]

PROGRAM CODE:

```
#include<iostream>
using namespace std;
void ChangeHigherValue(int& a, int& b)
{
   if(a > b)
   {
      a = -1;
   }
   else
   if(a < b)
   {
      b = -1;
   }
}
```

```
int main()
{
   int a, b;
   cout << "Input for two integers: ";
   cin >> a >> b;


   ChangeHigherValue(a, b);

   cout << "a = " << a << "\n";
   cout << "b = " << b << "\n";

   return 0;
}
```

OUTPUT:



4. Create a class student which stores name, date-of-birth and date-of-joining of a student. The data members date-of-birth and date-of-joining should be the objects of another class called 'date'. Input the data for 10 students and display it.

PROGRAM CODE:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;


class Student{
        string name;
        public:
                string dob;
                string doj;

                void input()
                {
```

```
                cout<<"Enter student name : ";
                string s;
                getline(cin,s);
                name = s;
                cout<<"Enter Date-of-Birth : ";
                getline(cin,dob);
                cout<<"Enter Date-of-Joining : ";
                getline(cin,doj);
        }

        void printName()
        {
                cout<<"Student name : "<<name<<endl;
        }
};


class Date: public Student
{
        public:
                void print()
                {
                        cout<<"Date-of-Birth : "<<dob<<endl;
                        cout<<"Date-of-Joining : "<<doj<<endl;
                }
};


int main()
{
        int n;
        cout<<"Enter details of 3 students : "<<endl<<endl;
        Date d[3];

        for(int i=0; i<3; i++)
        {
                d[i].input();
                cout<<endl;
        }


        cout<<endl<<"Displaying details of students : "<<endl;
        cout<<"--------------------------------"<<endl<<endl;
        for(int i=0; i<3; i++)
        {
                d[i].printName();
                d[i].print();
                cout<<endl;
        }
}
```

OUTPUT:
(3 students taken for purpose of ease)

```
Enter details of 3 students :

Enter student name : Anirban
Enter Date-of-Birth : 03 11 02
Enter Date-of-Joining : 04 05 23

Enter student name : Dushyant
Enter Date-of-Birth : 11 02 89
Enter Date-of-Joining : 04 09 07

Enter student name : Robin
Enter Date-of-Birth : 06 02 92
Enter Date-of-Joining : 22 05 24


Displaying details of students :
---------------------------------

Student name : Anirban
Date-of-Birth : 03 11 02
Date-of-Joining : 04 05 23

Student name : Dushyant
Date-of-Birth : 11 02 89
Date-of-Joining : 04 09 07

Student name : Robin
Date-of-Birth : 06 02 92
Date-of-Joining : 22 05 24
```

5. Write a program to demonstrate the order of call of constructors and destructors in case of multiple inheritance where one or more base classes are virtual.

PROGRAM CODE:

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

class Person
{
public:
  Person (int x)
  {

     cout << "Person(x) called" << endl;
  }
  Person ()
  {

     cout << "Person called" << endl;
  }
};
```

```
class Faculty:virtual public Person
{
   public:

  Faculty (int x):Person (x)
  {

    cout << "Faculty(x) called" << endl;
  }
};

class Student:virtual public Person
{

public:
  Student (int x):Person (x)
  {
    cout << "Student(x) called" << endl;
  }
};

class TA:public Faculty, public Student
{
public:
  TA (int x):Student (x), Faculty (x)
  {
    cout << "TA(x) called" << endl;
  }
};

int
main ()
{
  TA ta1 (30);
  return 0;
}
```

OUTPUT:

```
Person called
Faculty(x) called
Student(x) called
TA(x) called

--------------------------------
Process exited after 0.07749 seconds
Press any key to continue . . .
```

6. WAP to declare a class which stores a complex number. Demonstrate the use of constant objects , constant member function and constant arguments, using this class.

PROGRAM CODE:

```
#include <iostream>
#include <string>
#include <sstream>
class Complex
{
private:
        double x, y;
public:
Complex ()
  {
        x = 0;
        y = 0;
  }
Complex (double x, double y)
  {
        this->x = x;
        this->y = y;
  }
Complex operator + (Complex const &object)
  {

Complex re;
   re.x = x + object.x;
   re.y = y + object.y;
   return re;

}

Complex operator - (Complex const &object)        //use of constant objects
  {
   Complex re;
   re.x = x - object.x;
   re.y = y - object.y;
   return re;

}

std::string print() //demonstrating constant member function and constant arguments
  {

std::string result = "";

std::ostringstream x_sstream;

x_sstream << x;
```

```
std::string x_str = x_sstream.str ();

std::ostringstream y_sstream;

y_sstream << y;

std::string y_str = y_sstream.str ();

if (y < 0)

result = y_str + y_str + "i";

    else

result = x_str + "+" + x_str + "i";

return result;

}

};


int main ()
{

Complex number1 (3.5, 5.1),
number2 (-4.4, -3.8);


Complex number3 = number1 - number2;

std::cout << number3.print ();
   return 0;
}
```

OUTPUT: