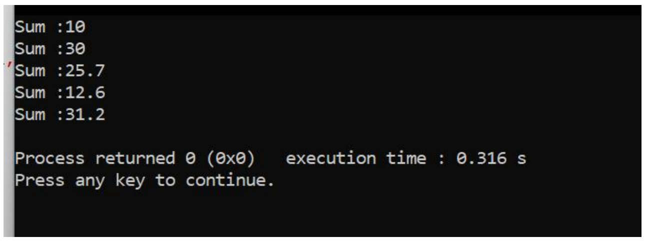
Module 4 [polymorphism]:

Topic 1 [Method/Function Overloading]
Problem Statement: Define a class Test where overload a method Sum() to sum numbers sent from main() function :
Theory :
Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.In c++ polymorphism is divided into two types:
1) Compile time polymorphism 2)Runtime polymorphism.
Function overloading is a compile time polymorphism. Function overloading means to have more than one function with the same name but with different parameters. Overloaded functions are differentiated by checking
1. Number of arguments. 2. Type & sequence of arguments but not by return type of the function

Code:

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

class Test{

public :
   int sum(int a, int b){
    return a+b;
   }
   double sum(double a,double b){
   return a+b;
   }

   int sum(int a){
   return a;
   }
    double sum(double a,int b){
   return a+b;
   }
    double sum(int a,double b){
   return a+b;
   }
};

int main(){

   Test t;

   cout<<"Sum :"<<t.sum(10)<<endl;
   cout<<"Sum :"<<t.sum(10,20)<<endl;
   cout<<"Sum :"<<t.sum(5.7,20)<<endl;
   cout<<"Sum :"<<t.sum(10,2.6)<<endl;
   cout<<"Sum :"<<t.sum(10.5,20.7)<<endl;

}
```

Output :



```
Sum :10
Sum :30
Sum :25.7
Sum :12.6
Sum :31.2

Process returned 0 (0x0)   execution time : 0.316 s
Press any key to continue.
```

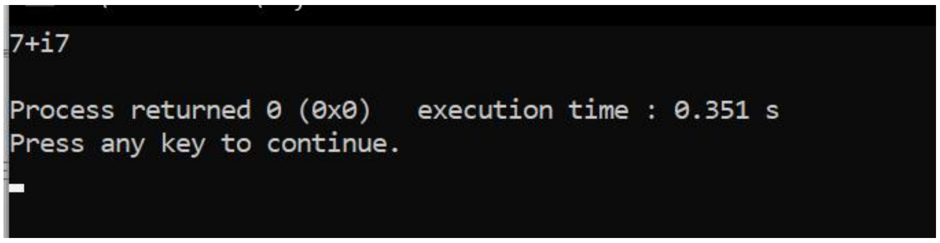
Topic 2 [Operator Overloading]:
<u>Problem Statement :</u> Suppose in a AC circuit, there are 3 impedances z1=3+j4, z2=4-j3 and z3=j6 are connected in Series. Now find the current in the circuit if input voltage is 100+j50. Implement operator overloading concept for your calculation. Use class Circuit and initialize the impedance values (real & img) by a constructor.

<u>Theory:</u>
In C++, we can make operators work for user-defined classes. This means C++ has the ability to provide the operators with a special meaning for a data type, this ability is known as operator overloading. For example, we can overload an operator '+' in a class like String so that we can concatenate two strings by just using +. Other example classes where arithmetic operators may be overloaded are Complex
Numbers, Fractional Numbers, Big Integer, etc. Operator overloading is a compile-time polymorphism. It is an idea of giving special meaning to an existing operator in C++ without changing its original meaning

<u>Code:</u>

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

class Circuit{

private:
int real;
int img;
public:
  Circuit(int r=0 , int i=0){
    real=r;
    img=i;
  }
  Circuit operator + (Circuit obj) {
  Circuit res;
  res.real = real + obj.real;
  res.img = img + obj.img;
  return res;
}

void print() {
cout<<real<<"+i"<<img<<endl;
}
};

int main(){
  Circuit z1(3,4);
  Circuit z2(4,-3);
  Circuit z3(0,6);

  Circuit z4=z1 + z2 + z3;

  z4.print();
}
```

Output :



```
7+i7

Process returned 0 (0x0)   execution time : 0.351 s
Press any key to continue.
```

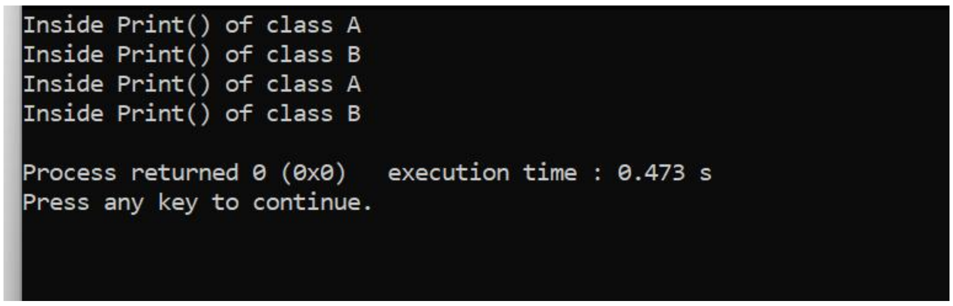
Discussion:
1.) Operator overloading is a compile-time polymorphism.So we have to be careful between compile time and run time polymorphism . 2.) We need to be careful between unary operator and binary operator .

Topic 3 [Method/Function Overriding]
Problem statement: Write a class A with a method Print() and a derived class B with method Print() overloaded. Now observe the output when following statements are written in the main() function-

Theoreitical background :
Function overridding provides multiple definitions of the function by changing signature i.e changing number of parameters, change datatype of parameters, return type doesn't play anyrole. It can be done in base as well as derived class.It is the redefinition of base class function in its derived class with same signature i.e return type and parameters. It can only be done in derived class.

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

class A{

public:
void virtual Print(){
cout<<"Inside Print() of class A"<<endl;
}
};
class B:public A{
public:
void Print(){
cout<<"Inside Print() of class B"<<endl;
}
};

int main(){

A a;
a.Print();
B b;
b.Print();

A *p;

p=&a;

p->Print();

p=&b;

p->Print();


}
```

Output :



```
Inside Print() of class A
Inside Print() of class B
Inside Print() of class A
Inside Print() of class B

Process returned 0 (0x0)   execution time : 0.473 s
Press any key to continue.
```

Topic 4 [Pure Virtual Function]
Problem statement: Modify the class defined in Topic 3 executes the following statements i)-iv) and observe the output:

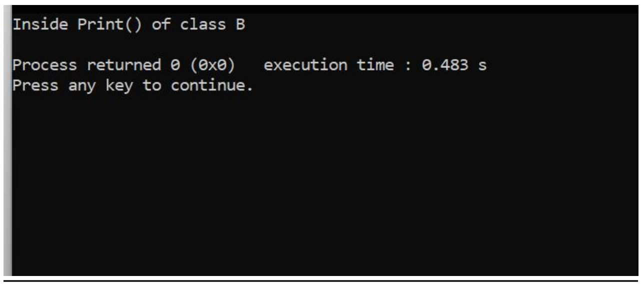
Theory:
Sometimes implementation of all function cannot be provided in a base class because we don't know the implementation. Such a class is called abstract class. For example, let Shape be a base class. We cannot provide implementation of function draw() in Shape, but we know every derived class must have implementation of draw(). Similarly an Animal class doesn't have implementation of move() (assuming that all animals move), but all animals must know how to move. We cannot create objects of abstract classes. A pure virtual function (or abstract function) in C++ is a virtual for which we can have implementation, But we must override that function in the derived class, otherwise the derived class will also become abstract class.

Code :

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

class A{
public:
virtual void Print()=0;
};

class B : public A{
public:
void Print(){
```

```
cout<<"Inside Print() of class B"<<endl;
  }
};

int main(){

B b;
A *p;
p=&b;

p->Print();

}
```

<u>Output :</u>



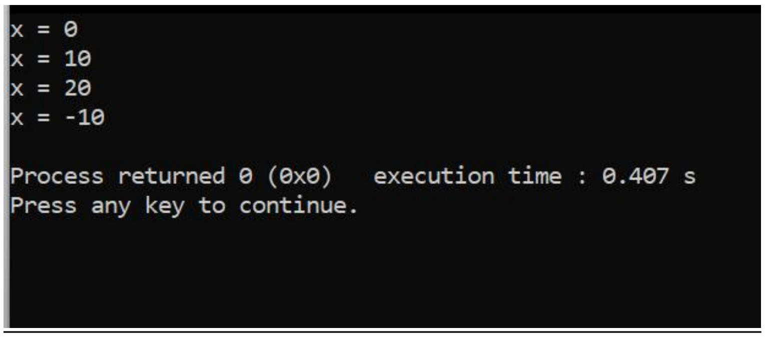
```
Inside Print() of class B

Process returned 0 (0x0)   execution time : 0.483 s
Press any key to continue.
```

Topic 5 [Friend Function]
<u>Problem Statement:</u> Using the following class, write three friend functions
i) Add(): Assign value to the data member x
ii) IncX() : Increase the value of x by m
iii) DecX() : Decrease the value of x by n

<u>Theory:</u>
Friend Class A friend class can access private and protected members of other class in which it is declared as friend. It is sometimes useful to allow a particular class to access private members of other class. For example, a LinkedList class may be allowed to access private members of Node. A friend class can access both private and protected members of the class in which it has been declared as friend.
<u>Code :</u>

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

class A{
private :
   int x;
public :
   A(){
    x=0;
   }
   void Display_data(){
      cout<<"x = "<< x <<endl;
   }
   friend void Add(A &value);
   friend void Incx(A &value);
   friend void Decx(A &value);
};

void Add(A &value){
   value.x=value.x + 10;
}
void Incx(A &value){
   value.x=value.x + 20;
}
void Decx(A &value){
   value.x=value.x - 10;
}

int main(){

   A val,inc,dec;
   val.Display_data();
   Add(val);
   val.Display_data();
   Incx(inc);
   inc.Display_data();
   Decx(dec);
   dec.Display_data();


}
```

Output :



```
x = 0
x = 10
x = 20
x = -10

Process returned 0 (0x0)   execution time : 0.407 s
Press any key to continue.
```

Discussion:
1.) We know friend function and friend class is not a same thing so need to be careful about it. 2.) Here I use friend function only to access the private variables .