EXPERIMENT NO-8

Title: Write a program to implement operator overloading for extraction operator >> and insertion operator << .

Objectives:

1. study of friend functions
2. implementation of operator overloading for extraction operator >> and insertion operator << .

Theory:

Friend Functions of Classes:

A friend function of a class is a non-member function of the class but has access to all of the members (public or non-public) of the class. To make a function be a friend to a class, the reserved word friend precedes the function prototype (in the class definition). The word friend appears only in the function prototype in the class definition, not in the definition of the friend function.

Friend Function Syntax:

```
class className {
    ... .. ...
    friend returnType functionName(arguments);
    ... .. ...
}
```

By using the keyword, the 'friend' compiler understands that the given function is a friend function.

Friend functions of the class are granted permission to access private and protected members of the class in C++. They are defined globally outside the class scope. Friend functions are not member functions of the class.

```
returnType  functionName(arguments){

}
```

Example of a friend function:

```
#include <iostream>
using namespace std;
class Box
{
   private:
        int length;
   public:
         Box (){
            length =5;
         }
   friend int incementLength (Box); //friend function
};
```

```
int incementLength (Box b)
{
    b. length +=10;
    return b.length;
}
int main ()
{
   Box b;
   cout <<” Length of box:” << incementLength(b)<<endl;
    return 0;
}
```

extraction >> and insertion << operator overload:

In C++, stream insertion operator “<<” is used for output and extraction operator “>>” is used for input.

following things must be known before overloading these operators.

1) cout is an object of ostream class and cin is an object istream class
2) These operators must be overloaded as a global function. And if we want to allow them to access private data members of class, we must make them friend.

Why these operators must be overloaded as global?
In operator overloading, if an operator is overloaded as member, then it must be a member of the object on left side of the operator. For example, consider the statement “ob1 + ob2” (let ob1 and ob2 be objects of two different classes). To make this statement compile, we must overload ‘+’ in class of ‘ob1’ or make ‘+’ a global function.
The operators ‘<<' and '>>' are called like 'cout << ob1' and 'cin >> ob1'. So if we want to make them a member method, then they must be made members of ostream and istream classes, which is not a good option most of the time. Therefore, these operators are overloaded as global functions with two parameters, cout and object of user defined class.

OVERLOADING THE STREAM INSERTION OPERATOR (<<)

The general syntax to overload the stream insertion operator, <<, for a class is described next.

Function Prototype (to be included in the definition of the class):

friend ostream& operator<< (ostream&, const className&);

Function Definition:

ostream& operator<<(ostream& osObject, const className& cObject)

{

    //local declaration, if any

    //Output the members of cObject.

    //osObject << . . .

```
        //Return the stream object.

        return osObject;

}
```

The extraction operator can be declared and defined in similar way.

Procedure:

1. Declare class for complex number.
2. Provide overloaded insertion operator to print complex number.
3. Provide overloaded extraction operator to store data for complex number.

Keywords:

Friend function, Cin, cout, insertion operator, extraction operator.