SE-201 Object Oriented Concepts and Programming

Name : Musadique Hussain

Roll No. : 31

Semester : Spring_____ Section: A

Group : Software Engineering

Date : 03/June/2022

Remarks : _____

Signature : _____

Lab #08

Operator Overloading

Exercise

1): Write any program in C++ that overloads Logical Not Operators (!). Following is the prototype of overload logical Not (!) operator:

bool operator! ()

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

class rectangle {
private:
    double length;
    double width;
public:
    rectangle() {
        length = 0;
        width = 0;
    }
    bool operator!=(const rectangle&)const;
    rectangle(double l, double w) {
        length = l;
        width = w;
    }
};
bool rectangle:: operator!=(const rectangle& rectangle) const
{
    return(length != rectangle.length || rectangle.width);
}
int main() {
    rectangle r1(2.3, 4.3);
    rectangle r2(22.3, 4.3);
    if (r1 != r2) {
        cout << "They are not equal";

    }
    else {
        cout << "They are equal";
    }
    system(_Command:"pause>0");
    return 0;
}
```

```
E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
They are not equal
```

2) Define a class named A, with member variables int x and float y, overload a binary operator (+), add two objects of class A and return their sum in 3rd object.

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

class rectangle {
private:
    int num;
    float deci;
public:
    rectangle() {
        num = 0;
        deci = 0;
    }
    rectangle operator+(const rectangle&) const;
    rectangle(int x, float y) {
        num = x;
        deci = y;
    }
    void getdata() {
        cout << "Value of num is " << num << endl;
        cout << endl;
        cout << "Value of deci is " << deci << endl;
        cout << endl;
        cout << "The sum of both is " << num + deci << endl;
        cout << endl;
    }
};
rectangle rectangle::operator+(const rectangle& rectangletype)const {

    rectangle add;
    add.num = num + rectangletype.num;
    add.deci = deci + rectangletype.deci;
    return add;
}
int main() {
    rectangle r1(4, 2.5);
    rectangle r2(5, 1.5);
    r1.getdata();
    r2.getdata();
    system("pause>0");

}
```

```
E:\Study Materials\C++\sasageyo\x64\Debug\sasageyo.exe
Value of num is 4

Value of deci is 2.5

The sum of both is 6.5

Value of num is 5

Value of deci is 1.5

The sum of both is 6.5
```

3) Write a program to overload the insertion (<<) and Extraction (>>) operator.

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

class Complex
{
private:
    int real;
    int imag;
public:
    Complex(int r = 0, int i = 0)
    {
        real = r;
        imag = i;
    }
    friend ostream& operator<<(ostream& out, const  Complex& c);
    friend istream& operator>>(istream& in, Complex& c);
};
ostream& operator<<(ostream& out, const Complex& c)
{
    out << c.real;
    out << "+i" << c.imag << endl;
    return out;
}
istream& operator>>(istream& in, Complex& c)
{
    cout << "Enter real part:";
    in >> c.real;
    cout << "Enter imaginary part:";
    in >> c.imag;
    return in;
}

int main(int argc, char** argv) {
    Complex c1;
    cin >> c1;
    cout << "The complex object is:";
    cout << c1;
    system("pause>0");
    return 0;
}

```

```
E:\Study Materials\C++\Ben\x64\Debug\Ben.exe
Enter real part:2
Enter imaginary part:5
The complex object is:2+i5
```

4) Define a class abc with data members a and b. Define a method to overload unary operator (++) for the class objects.

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

class ADD {
private:
    int n;
public:
    ADD() {
        n = 0;
    }
    void show() {
        cout << "The value of n is " << n << endl;
    }
    void showdata() {
        cout << "The value of n after increment is " << n << endl;
    }
    void operator ++() {
        n = n + 1;
    }
};
int main() {
    ADD num;
    num.show();
    ++num;
    num.showdata();
    system("pause>0");
    return 0;
}
```