Assignment - 27
Operator overloading and friend function

1. Define a class Complex with appropriate instance variables and member functions. Define following operators in the class:

a. +
b. -
c. *
d. ==

Program -

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

class Complex
{
    private:
       int real, imag;

    public:
        void setData(int r, int i)
        {
            real = r;
            imag = i;
        }
        void showData()
        {
            cout<<"Real = "<<real<<", "<<"Imaginary =
      "<<imag<<endl;
        }
        Complex operator+(Complex C)
        {
            Complex temp;
            temp.real = real + C.real;
            temp.imag = imag + C.imag;
            return temp;
        }
        Complex operator-(Complex C)
        {
            Complex temp;
            temp.real = real - C.real;
            temp.imag = imag - C.imag;
            return temp;
        }
        Complex operator*(Complex C)
        {
            Complex temp;
            temp.real = real * C.real;
            temp.imag = imag * C.imag;
            return temp;
        }
```

```
        int operator==(Complex C)
        {
           if(real == C.real && imag == C.imag)
                return 1;
            else
                return 0;
        }
};
int main()
{
    Complex c1,c2,c3;
    c1.setData(5,7);
    c2.setData(5,7);

    c1.showData();
    c2.showData();
    cout<<"Addition-\n";
    c3 = c1 + c2; // Addition of objects
    c3.showData();

    cout<<"Subtraction-\n";
    c3 = c1 - c2; // Subtraction of objects
    c3.showData();

    cout<<"Multiplication-\n";
    c3 = c1 * c2; // Product of objects
    c3.showData();

    cout<<"Checking equality-\n";
    if(c1 == c2) // Checking equality of objects
        cout<<"Objects are equal";
    else
        cout<<"Objects are not equal";
    return 0;
}
```

Output -

Real = 5, Imaginary = 7
Real = 5, Imaginary = 7
Addition-
Real = 10, Imaginary = 14
Subtraction-
Real = 0, Imaginary = 0
Multiplication-
Real = 25, Imaginary = 49
Checking equality-
Objects are equal

---

2. Write a C++ program to overload unary operators that is increment and decrement.

Program -

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

class Overload
{
    private:
        int x;
    public:
        void setData(int a)
        {
            x = a;
        }
        void operator++() // function to overload prefix ++
        {
            ++x;
        }
        void operator++(int) // function to overload postfix ++
        {
            x++;
        }
        void operator--() // function to overload prefix ++
        {
            --x;
        }
        void operator--(int) // function to overload postfix ++
        {
            x--;
        }
        void showData()
        {
            cout<<"x = "<<x<<endl;
        }
};
int main()
{
    Overload obj;
    obj.setData(8);
    ++obj;
    obj.showData();
    obj--;
    obj.showData();
    --obj;
    obj.showData();
    return 0;
}
```

Output -
x = 9
x = 8
x = 7

---

3. Write a C++ program to add two complex numbers using operator overloaded by a friend function.

Program -

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

class Complex
{
    private:
        int real, imag;

    public:
        void setData(int r, int i)
        {
            real = r;
            imag = i;
        }
        void showData()
        {
            cout<<"Real = "<<real<<", "<<"Imaginary =
      "<<imag<<endl;
        }
        friend Complex operator+(Complex, Complex);
};

Complex operator+(Complex X, Complex Y)
{
    Complex temp;
    temp.real = X.real + Y.real;
    temp.imag = X.imag + Y.imag;
    return temp;
}

int main()
{
    Complex c1, c2, c3;

    c1.setData(7,2);
    c2.setData(9,9);

    c1.showData();
    c2.showData();

    c3 = c1 + c2;
    cout<<"After adding -\n";
```

```
    c3.showData();
    return 0;
}
```

Output -
Real = 7, Imaginary = 2
Real = 9, Imaginary = 9
After adding -
Real = 16, Imaginary = 11

---

4. Create a class Time which contains:
- Hours
- Minutes
- Seconds
Write a C++ program using operator overloading for the following:
1. = = : To check whether two Times are the same or not.
2. >> : To accept the time.
3. << : To display the time.

Program -

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

class Time
{
    private:
        int hour, min, sec;
    public:
        friend istream& operator>>(istream&, Time&);
        friend ostream& operator<<(ostream&, Time&);
        friend int operator==(Time, Time);
};

istream& operator>>(istream &input, Time &T)
{
    cout<<"Enter Hours   :   ";
    input>>T.hour;
    cout<<"Enter Minutes :   ";
    input>>T.min;
    cout<<"Enter Seconds :   ";
    input>>T.sec;
    return input;
}

ostream& operator<<(ostream &output, Time &T)
{
    output<<"Hours   :  "<<T.hour<<endl;
    output<<"Minutes :  "<<T.min<<endl;
    output<<"Seconds :  "<<T.sec<<endl;
    return output;
}
```

```
int operator==(Time T1, Time T2)
{
    if(T1.hour == T2.hour && T1.min == T2.min && T1.sec == T2.sec)
        return 1;

    return 0;
}

int main()
{
    Time t1, t2;

    cout<<"Enter First time\n--------------------\n";
    cin>>t1;
    cout<<"\nFirst Time\n";
    cout<<t1;
    cout<<"\nEnter Second time\n--------------------\n";
    cin>>t2;
    cout<<"\nSecond Time\n";
    cout<<t2;
    if(t1 == t2)
        cout<<"\nTimes are same";
    else
        cout<<"\nTimes are not same";
    return 0;
}
```

Output -

```
Enter First time
------------------------
Enter Hours     :  12
Enter Minutes   :  34
Enter Seconds   :  2

First Time
Hours   : 12
Minutes :  34
Seconds :  2

Enter Second time
----------------------------
Enter Hours     :  12
Enter Minutes   :  12
Enter Seconds   :  3

Second Time
Hours     : 12
Minutes   : 12
Seconds   : 3

Times are not same
```

5. Consider following class Numbers

```
class Numbers
{
int x,y,z;
public:
// methods
};
```

Overload the operator unary minus (-) to negate the numbers.

Program -

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

class Numbers
{
    private:
        int x, y, z;

    public:
        void setData(int a, int b, int c)
        {
            x = a;
            y = b;
            z = c;
        }
        void showData()
        {
            cout<<"x = "<<x<<", "<<"y = "<<y<<", "<<"z =
"<<z<<endl;
        }
        Numbers operator-()
        {
            Numbers temp;
            temp.x = -x;
            temp.y = -y;
            temp.z = -z;
            return temp;
        }
};

int main()
{
    Numbers n1;
    n1.setData(13, 3, 4);
    n1.showData();
    n1 = -n1;
    cout<<"Negative-\n";
    n1.showData();
    return 0;
}
```

Output -
x = 13, y = 3, z = 4
Negative-
x = -13, y = -3, z = -4

---

6. Create a class CString to represent a string.
a) Overload the + operator to concatenate two strings.
b) == to compare 2 strings.

Program -

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

class CString
{
    private:
        char str[100];
    public:
        void setString(const char ch[])
        {
            strcpy(str,ch);
        }
        void showString()
        {
            cout<<str;
        }

        CString operator+(CString S)
        {
            CString temp;
            strcpy(temp.str,str);
            strcat(temp.str,S.str);
            return temp;
        }
        int operator==(CString S)
        {
            if(strcmp(str,S.str) == 0)
                return 1;
            return 0;
        }
        void length()
        {
            cout<<"\nLength of the string is: "<<strlen(str);
        }
};

int main()
{
    CString str1, str2, str3;

    str1.setString("Hello");
    cout<<"String number - 1 : ";
```

```
    str1.showString();
    str1.length();

    str2.setString("Hello");
    cout<<"\nString number - 2 : ";
    str2.showString();
    str2.length();

    str3 = str1 + str2;
    cout<<"\nConcatenated string is: ";
    str3.showString();
    str3.length();
     if(str1 == str2)
        cout<<"\nString number 1 and 2 are equal";
    else
        cout<<"\nString number 1 and 2 are not equal";
    return 0;
}
```

Output -
String number - 1 : Hello
Length of the string is: 5
String number - 2 : Hello
Length of the string is: 5
Concatenated string is: HelloHello
Length of the string is: 10
String number 1 and 2 are equal

---

7. Define a C++ class fraction
class fraction
{
long numerator;
long denominator;
Public:
fraction (long n=0, long d=0);
}
Overload the following operators as member or friend:
a) Unary ++ (pre and post both)
b) Overload as friend functions: operators << and >>.

Program -

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

class Fraction
{
    private:
        long numerator;
        long denominator;
```

```
    public:
        Fraction(long n=0, long d=0)
        {
            numerator = n;
            denominator = d;
        }

        friend ostream& operator<<(ostream&, Fraction&);
        friend istream& operator>>(istream&, Fraction&);

        Fraction operator++()
        {
            Fraction temp;
            temp.numerator = ++numerator;
            temp.denominator = ++denominator;
            return temp;
        }

        Fraction operator++(int dummy)
        {
            Fraction temp;
            temp.numerator = numerator++;
            temp.denominator = denominator++;
            return temp;
        }
};

ostream& operator<<(ostream &out, Fraction &F)
{
    out<<F.numerator<<"/"<<F.denominator<<endl;
    return out;
}

istream& operator>>(istream &in, Fraction &F)
{
    cout<<"Enter numerator   :  ";
    in>>F.numerator;
    cout<<"Enter denominator :  ";
    in>>F.denominator;
    return in;
}

int main()
{
    Fraction f1, f2;

    cout<<"f1   :  "<<f1;
    cout<<"f2   :  "<<f2;

    cout<<"\nEnter 1st Fraction value"<<endl;
    cin>>f1;

    f1++;
```

```
    cout<<"\nf1++   :  "<<f1;

    ++f1;
    cout<<"++f1   :  "<<f1;

    cout<<"\nEnter 2nd Fraction value\n";
    cin>>f2;

    f2 = ++f1;
    cout<<"\nf2 = ++f1"<<endl;

    cout<<"f1   :  "<<f1;
    cout<<"f2   :  "<<f2;

    f2 = f1++;
    cout<<"\nf2 = f1++"<<endl;

    cout<<"f1   :  "<<f1;
    cout<<"f2   :  "<<f2;
    return 0;
}
```

Output -

f1 : 0/0
f2 : 0/0

Enter 1st Fraction value
Enter numerator : 2
Enter denominator : 3

f1++ : 3/4
++f1 : 4/5

Enter 2nd Fraction value
Enter numerator : 1
Enter denominator : 2

f2 = ++f1
f1 : 5/6
f2 : 5/6

f2 = f1++
f1 : 6/7
f2 : 5/6

---

8. Consider a class Matrix
Class Matrix
{
int a[3][3];
Public:
//methods;
};
Overload the - (Unary) should negate the numbers stored in the object.

Program -

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

class Matrix
{
    private:
        int a[3][3];

    public:
        void inputMatrix()
        {
            int i, j;
            for(i = 0; i < 3; i++)
            {
                for(j = 0 ; j < 3; j++)
                {
                    cin>>a[i][j];
                }
            }
        }

        void showMatrix()
        {
            int i, j;
            for(i = 0; i < 3; i++)
            {
                for(j = 0 ; j < 3; j++)
                {
                    cout<<a[i][j]<<"      ";
                }
                cout<<endl;
            }
        }

        Matrix operator-()
        {
            Matrix temp;
            int i, j;
            for(i = 0; i < 3; i++)
            {
                for(j = 0; j < 3; j++)
                {
                    temp.a[i][j] = -a[i][j];
                }
            }
            return temp;
        }
};

int main()
{
    Matrix mat;
```

```
    cout<<"Enter Matrix element (3 x 3) :\n";
    mat.inputMatrix();
    cout<<"\nMatrix is :\n\n";
    mat.showMatrix();
    mat = -mat;
    cout<<"\nMatrix is :\n\n";
    mat.showMatrix();
    return 0;
}
```

Output -

Enter Matrix element (3 x 3) :
1 2 3 4 5 6 7 8 9

Matrix is :

1 2 3
4 5 6
7 8 9

Matrix is :

-1 -2 -3
-4 -5 -6
-7 -8 -9

---

9. Consider the following class mystring
Class mystring
{
char str [100];
Public:
// methods
};
Overload operator "!" to reverse the case of each alphabet in the string (Uppercase to Lowercase and vice versa).

Program -

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

class MyString
{
    private:
        char str[100];

    public:
        void setString(const char ch[])
        {
            strcpy(str,ch);
        }
```

```
        void showString()
        {
            cout<<str<<endl;
        }
        MyString operator!()
        {
            MyString temp;
            int i;

            strcpy(temp.str, str);
            for(i = 0; temp.str[i]; i++)
            {
                if(temp.str[i] >= 'a' && temp.str[i] <= 'z')
                    temp.str[i] = temp.str[i] - 32;
                else if(temp.str[i] >= 'A' && temp.str[i] <= 'Z')
                    temp.str[i] = temp.str[i] + 32;
            }
            return temp;
        }
};

int main()
{
    MyString str;
    str.setString("MySirG");
    str.showString();
    str = !str;
    cout<<"After reversing the case of each alphabet -\n";
    str.showString();
    return 0;
}
```

Output -
MySirG
After reversing the case of each alphabet -
mYsIRg

10.Class Matrix
{
int a[3][3];
Public:
//methods;
};
Let m1 and m2 are two matrices. Find out m3=m1+m2 (use operator overloading).

Program -

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

class Matrix
{
    private:
        int a[3][3];

    public:
        void inputMatrix()
        {
            int i, j;
            for(i = 0; i < 3; i++)
            {
                for(j = 0; j < 3; j++)
                {
                    cin>>a[i][j];
                }
            }
        }
        void showMatrix()
        {
            int i, j;
            for(i = 0; i < 3; i++)
            {
                for(j = 0; j < 3; j++)
                {
                    cout<<a[i][j]<<"    ";
                }
                cout<<endl;
            }
        }
        Matrix operator+(Matrix M)
        {
            Matrix temp;
            int i, j, sum;

            for(i = 0; i < 3; i++)
            {
                for(j = 0; j < 3; j++)
                {
                    temp.a[i][j] = a[i][j] + M.a[i][j];
                }
            }
            return temp;
        }
};

int main()
{
    Matrix m1, m2, m3;
```

```
    cout<<"Enter First Matrix elements (3 x 3) :\n";
    m1.inputMatrix();
    cout<<"Enter Second Matrix elements (3 x 3) :\n";
    m2.inputMatrix();

    cout<<"\nFirst Matrix :\n";
    m1.showMatrix();
    cout<<"\nSecond Matrix :\n";
    m2.showMatrix();

    m3 = m1 + m2;
    cout<<"\nAddition of Matrix :\n";
    m3.showMatrix();

    return 0;
}
```

Output -

```
Enter First Matrix elements (3 x 3) :
1 2 3 4 5 6 7 8 9
Enter Second Matrix elements (3 x 3) :
8 6 4 2 0 1 1 0 0

First Matrix :
1   2   3
4   5   6
7   8   9

Second Matrix :
8   6   4
2   0   1
1   0   0

Addition of Matrix :
9   8   7
6   5   7
8   8   9
```
