**Assignment-27 Solution Name: Om Pant**

**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. ==

Ans-

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

// a. +

// b. -

// c. \*

// d. ==

#include<iostream>

using namespace std;

class Complex{

    int real,img;

    public:

        void setvalues(){

            cout<<"Enter Real Part: ";

            cin>>real;

            cout<<"Enter Imaginary Part: ";

            cin>>img;

        }

        void printValues(){

            if(img<0){

                cout<<real<<img<<'i'<<endl;

            }else{

                cout<<real<<" + "<<img<<'i'<<endl;

            }

        }

        Complex operator+(Complex c){

            Complex temp;

            temp.real = real + c.real;

            temp.img = img + c.img;

            return temp;

        }

        Complex operator-(Complex c){

            Complex temp;

            temp.real = real - c.real;

            temp.img = img - c.img;

            return temp;

        }

        Complex operator\*(Complex c){

            Complex temp;

            temp.real =((real \* c.real ) - (img \* c.img));

            temp.img = ((real \* c.img) + (img \* c.real));

            return temp;

        }

        int operator==(Complex c){

            if(real == c.real && img == c.img){

                return 1;

            }

            else{

                return 0;

            }

        }

};

int main(){

    Complex c1,c2,c3;

    int x;

    c1.setvalues();

    c2.setvalues();

    cout<<"Two Complex numbers are: "<<endl;

    c1.printValues();

    c2.printValues();

    c3 = c1 + c2;

    cout<<"Addition:"<<endl;

    c3.printValues();

    cout<<"Substraction:"<<endl;

    c3 = c1 - c2;

    c3.printValues();

    cout<<"Multiplication:"<<endl;

    c3 = c1 \* c2;

    c3.printValues();

    cout<<"Are Entered Complex Numbers Equal: ";

    x = c1==c2;

    if(x)

        cout<<" YES ";

    else

        cout<<" NO ";

    return 0;

}

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

Ans-

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

//ans- Here i'm overloading ++ and -- operator for complex numbers

#include<iostream>

using namespace std;

class Complex{

    int real,img;

    public:

        void setvalues(){

            cout<<"Enter Real Part: ";

            cin>>real;

            cout<<"Enter Imaginary Part: ";

            cin>>img;

        }

        void printValues(){

            if(img<0){

                cout<<real<<img<<'i'<<endl;

            }else{

                cout<<real<<" + "<<img<<'i'<<endl;

            }

        }

        void operator++(int){

            real++;

            img++;

        }

        void operator--(int){

           real--;

           img--;

        }

};

int main(){

    Complex c1;

    int x;

    c1.setvalues();

    cout<<"Complex number entered: "<<endl;

    c1.printValues();

    cout<<"Incrementing Complex number  :";

    c1++;

    c1.printValues();

    cout<<"Decrementing Complex number  :";

    c1--;

    c1.printValues();

    return 0;

}

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

Ans-

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

#include<iostream>

using namespace std;

class Complex{

    int real,img;

    public:

        void setvalues(){

            cout<<"Enter Real Part: ";

            cin>>real;

            cout<<"Enter Imaginary Part: ";

            cin>>img;

        }

        void printValues(){

            if(img<0){

                cout<<real<<img<<'i'<<endl;

            }else{

                cout<<real<<" + "<<img<<'i'<<endl;

            }

        }

        friend Complex operator+(Complex,Complex);

};

Complex operator+(Complex a,Complex b){

    Complex temp;

    temp.real = a.real + b.real;

    temp.img = a.img +  b.img;

    return temp;

}

int main(){

    Complex c1,c2,c3;

    int x;

    c1.setvalues();

    c2.setvalues();

    cout<<"Two Complex numbers are: "<<endl;

    c1.printValues();

    c2.printValues();

    c3 = c1 + c2;

    cout<<"Addition:"<<endl;

    c3.printValues();

    return 0;

}

4. Create a class Time which contains:

- Hours

- Minutes

- Seconds

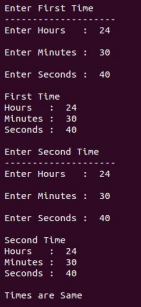
Write a C++ program using operator overloading for the following:

1. = = : To check whether two Times are the same or not.

2. >> : To accept the time.

3. << : To display the time.

Output -



Ans-

// 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.

// Output -

// image.png

#include<iostream>

using namespace std;

class Time{

    int hour,minute,second;

    public:

        friend ostream& operator<<(ostream &s, Time t){

            cout<<"Hours   : "<<t.hour<<endl;

            cout<<"Minutes : "<<t.minute<<endl;

            cout<<"Seconds : "<<t.second<<endl;

            return s;

        }

        friend istream& operator>>(istream &i, Time &t){

            cout<<"Enter Hours: ";

            cin>>t.hour;

            cout<<"Enter Minutes: ";

            cin>>t.minute;

            cout<<"Enter Seconds: ";

            cin>>t.second;

            return i;

        }

         bool operator==(Time x){

            if(hour == x.hour && minute == x.minute && second == x.second){

                return true;

            }

            else{

                return false;

            }

        }

};

int main(){

    Time a,b;

    cout<<"Enter First Time: "<<endl;

    cout<<"-----------------"<<endl;

    cin>>a;

    cout<<"First Time:"<<endl;

    cout<<a;

    cout<<"Enter Second Time: "<<endl;

    cout<<"-----------------"<<endl;

    cin>>b;

    cout<<"Second Time:"<<endl;

    cout<<b;

    if(a == b){

        cout<<"Times are same"<<endl;

    }

    else{

        cout<<"Times are Different"<<endl;

    }

    return 0;

}

5. Consider following class Numbers

*class Numbers*

*{*

*int x,y,z;*

*public:*

*// methods*

*};*

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

// 5. Consider following class Numbers

// class Numbers

// {

//  int x,y,z;

//  public:

//  // methods

// };

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

#include<iostream>

using namespace std;

class Numbers{

    int x,y,z;

    public:

        void setValues(){

            int a,b,c;

            cout<<"Enter 1st Value: ";

            cin>>a;

            cout<<"Enter 2nd Value: ";

            cin>>b;

            cout<<"Enter 3rd Value: ";

            cin>>c;

            x = a;

            y = b;

            z = c;

        }

        void printValues(){

            cout<<x<<" "<<y<<" "<<z<<endl;

        }

        void operator-(){

            x = -x;

            y = -y;

            z = -z;

        }

};

int main(){

    Numbers n;

    n.setValues();

    cout<<"Values Before unary minus: ";

    n.printValues();

    -n;

    cout<<"Values after unary minus: ";

    n.printValues();

    return 0;

}

6. Create a class CString to represent a string.

a) Overload the + operator to concatenate two strings. b) == to compare 2 strings.

Ans-

// 6. Create a class CString to represent a string.

//  a) Overload the + operator to concatenate two strings.  b) == to compare 2 strings.

#include<iostream>

#include<cstring>

using namespace std;

class CString{

    char str[50] = "";

    public:

        void input(){

            cout<<"Enter a string"<<endl;

            cin.getline(str,50);

        }

        void print(){

            cout<<str<<endl;

        }

        CString operator+(CString s){

            CString x;

            strcpy(x.str,str);

            strcat(x.str,s.str);

            return x;

        }

        bool operator==(CString s){

            if((strcmp(str, s.str)) == 0){

                return true;

            }

            else{

                return false;

            }

        }

};

int main(){

    CString str1,str2,str3;

    str1.input();

    str2.input();

    cout<<"Strings are.."<<endl;

    str1.print();

    str2.print();

    str3 = str1 + str2;

    str3.print();

    if(str1 == str2){

        cout<<"Strings are Same"<<endl;

    }

    else{

        cout<<"Strings are Different"<<endl;

    }

    return 0;

}

7. Define a C++ class fraction

*class fraction*

*{*

*long numerator;*

*long denominator;*

*Public:*

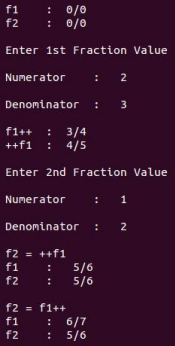
*fraction (long n=0, long d=0);*

*}*

Overload the following operators as member or friend:

a) Unary ++ (pre and post both)

b) Overload as friend functions: operators << and >>. Output-



Ans-

// 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 >>. Output-

// image.png

#include<iostream>

using namespace std;

class fraction

{

    long numerator;

    long denominator;

        public:

           fraction (long n=0, long d=0){

           }

            void operator++(void){

                ++numerator;

                ++denominator;

            }

            void operator++(int){

                numerator++;

                denominator++;

            }

           friend ostream& operator<<(ostream &o, fraction f){

            cout<<f.numerator<<"/"<<f.denominator;

            }

            friend istream& operator>>(istream &i, fraction &f){

                cout<<"Enter numerator: ";

                cin>>f.numerator;

                cout<<"Enter Denominator: ";

                cin>>f.denominator;

            }

};

int main(){

    fraction f1,f2;

    cout<<"Enter 1st Fraction: "<<endl;

    cin>>f1;

    cout<<"Enter 2nd Fraction: "<<endl;

    cin>>f2;

    cout<<"First Fraction: "<<endl<<f1<<endl<<"Second Fraction: "<<endl<<f2<<endl;

    cout<<"F1++ : " ;

    f1++;

    cout<<f1<<endl;

    cout<<"++F2 : ";

    ++f2;

    cout<<f2<<endl;

    return 0;

}

8. Consider a class Matrix

*Class Matrix*

*{*

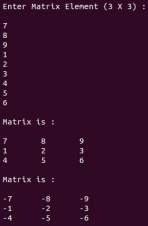
*int a[3][3];*

*Public:*

*//methods;*

*};*

Overload the - (Unary) should negate the numbers stored in the object. Output -



Ans-// 8. Consider a class Matrix

// Class Matrix

// {

//  int a[3][3];

//  Public:

//  //methods;

// };

// Overload the - (Unary) should negate the numbers stored in the object. Output -

// image.png

#include<iostream>

using namespace std;

class Matrix{

    int a[3][3];

    public:

        void inputMatrix(){

            for(int i=0;i<3;i++){

                for(int j=0;j<3;j++){

                    cin>>a[i][j];

                }

            }

        }

        void print(){

            for(int i=0;i<3;i++){

                for(int j=0;j<3;j++){

                    cout<<a[i][j]<<"  ";

                }

                cout<<endl;

            }

        }

        void operator-(){

            for(int i=0;i<3;i++){

                for(int j=0;j<3;j++){

                    a[i][j] \*= -1;

                }

            }

        }

};

int main(){

    Matrix m;

    cout<<"Enter Matrix Element: "<<endl;

    m.inputMatrix();

    cout<<"Matrix Element are: "<<endl;

    m.print();

    -m;

    cout<<"Enter Matrix Element: "<<endl;

    m.print();

}

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).

*Ans-*

// 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).

#include<iostream>

#include<cstring>

using namespace std;

class mystring

{

    char str[100];

    public:

        void input(){

            cout<<"Enter a string: ";

            cin.getline(str, 100);

        }

        void print(){

            cout<<"String: "<<str;

        }

        mystring operator!(){

            mystring temp;

            strcpy(temp.str,str);

            for(int i=0; temp.str[i]!= '\0';i++){

                if(temp.str[i]>=65 && temp.str[i]<=90){

                    temp.str[i] += 32;

                }

                else if(temp.str[i]>=97 && temp.str[i]<= 122){

                    temp.str[i] -= 32;

                }

            }

            return temp;

        }

};

int main(){

    mystring str1, str2;

    str1.input();

    cout<<"Entered String:  "<<endl;

    str1.print();

    str2 = !str1;

    cout<<"Transformed String: "<<endl;

    str2.print();

    return 0;

}

*10. Class Matrix*

*{*

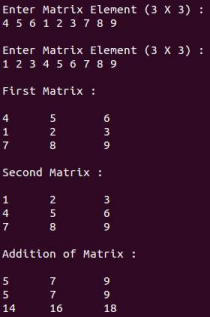
*int a[3][3];*

*Public:*

*//methods;*

*};*

Let m1 and m2 are two matrices. Find out m3=m1+m2 (use operator overloading). Output -



// 10. Class Matrix

// {

//  int a[3][3];

//  Public:

//  //methods;

// };

// Let m1 and m2 are two matrices. Find out m3=m1+m2 (use operator overloading). Output -

// image.png

#include<iostream>

using namespace std;

class Matrix{

    int a[3][3];

    public:

        void inputMatrix(){

            for(int i=0;i<3;i++){

                for(int j=0;j<3;j++){

                    cin>>a[i][j];

                }

            }

        }

        void print(){

            for(int i=0;i<3;i++){

                for(int j=0;j<3;j++){

                    cout<<a[i][j]<<"  ";

                }

                cout<<endl;

            }

        }

        Matrix operator+(Matrix m){

            Matrix temp;

            for(int i=0;i<3;i++){

                for(int 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 1st Matrix Elements: "<<endl;

    m1.inputMatrix();

    cout<<"Enter 2nd Matrix Elements: "<<endl;

    m2.inputMatrix();

    cout<<"Matrix 1  Elements are: "<<endl;

    m1.print();

    cout<<"Matrix 2  Elements are: "<<endl;

    m2.print();

    cout<<"Addition of matrices : "<<endl;

    m3 = m1 + m2;

    m3.print();

    return 0;

}