#include<iostream>

using namespace std;

class unaryTest {

private:

int x, y, z;

public:

void get(int a, int b , int c)

{

x = a;

y = b;

z = c;

}

void display();

void operator -();

};

void unaryTest::display()

{

cout << x << y << z;

}

void unaryTest::operator-()

{

x = -x;

y = -y;

z = -z;

}

int main()

{

unaryTest obj1;

obj1.get(5,10,15);

obj1.display();

-obj1;

obj1.display();

getchar();

getchar();

}

//////////////////////////////////////////////////////////////

#include <iostream>

using namespace std;

class Distance {

private:

int feet; // 0 to infinite

int inches; // 0 to 12

public:

// required constructors

Distance() {

feet = 0;

inches = 0;

}

Distance(int f, int i) {

feet = f;

inches = i;

}

// method to display distance

void displayDistance() {

cout << "F: " << feet << " I:" << inches << endl;

}

// overloaded minus (-) operator

Distance operator- ()

{

feet = -feet;

inches = -inches;

return Distance(feet, inches);

}

};

int main() {

Distance D1(11, 10), D2(-5, 11);

-D1; // apply negation

D1.displayDistance(); // display D1

-D2; // apply negation

D2.displayDistance(); // display D2

getchar();

getchar();

}

///////////////////////////////////////////////

#include<iostream>

using namespace std;

class IncreDecre

{

int a, b;

public:

void accept()

{

cout << "\n Enter Two Numbers : \n";

cout << " ";

cin >> a;

cout << " ";

cin >> b;

}

void operator--() //Overload Unary Decrement

{

a--;

b--;

}

void operator++() //Overload Unary Increment

{

a++;

b++;

}

void display()

{

/\*cout << "\n A : " << a;

cout << "\n B : " << b;\*/

cout << a << "+\t" << b << "i" << endl;

}

};

int main()

{

IncreDecre id;

id.accept();

--id;

cout << "\n After Decrementing : ";

id.display();

++id;

++id;

cout << "\n\n After Incrementing : ";

id.display();

getchar();

getchar();

}

/////////////////////////////////////////////////////////////////////

#include <iostream>

using namespace std;

class Check

{

private:

int i;

public:

Check() : i(0) { }

Check operator ++ ()

{

Check temp;

temp.i = ++i;

return temp;

}

// Notice int inside barcket which indicates postfix increment.

Check operator ++ (int)

{

Check temp;

temp.i = i++;

return temp;

}

void Display()

{

cout << "i = " << i << endl;

}

};

int main()

{

Check obj, obj1;

obj.Display();

obj1.Display();

// Operator function is called, only then value of obj is assigned to obj1

obj1 = ++obj;

obj.Display();

obj1.Display();

//Assigns value of obj to obj1, only then operator function is called.

obj1 = obj++;

obj.Display();

obj1.Display();

getchar();

getchar();

}