//#include<iostream>

//using namespace std;

//class complex

//{

//

//private:

// int real;

// int imag;

//

//public:

// complex()

// {}

//

// complex(int a,int b)

// {

// real=a;

// imag=b;

//

// }

//

// void operator + (complex c);

//

//

//};

//void complex::operator+(complex c)

//{

// complex temp;

// temp.real = real + c.real;

// temp.imag = imag + c.imag;

// cout << "real" << temp.real << endl;

// cout << "imag" << temp.imag << endl;

//

//

//}

//

//int main()

//{

// complex c1(10, 20);

// complex c2(30, 40);

// c1 + c2;

// getchar();

// getchar();

//

//}

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

//#include<iostream>

//using namespace std;

//class complex

//{

//

//private:

// int real;

//

//

//public:

//

//

// complex(int a)

// {

// real=a;

//

//

// }

//

// void operator + (complex a)

// {

// cout << "this is bainary operator";

// }

//

//

//};

//

//

//int main()

//{

// complex c1(10);

// complex c2(30);

//

// //c1 + c2;

//

// c1.operator+(c2);

//

// getchar();

// getchar();

//

//}

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

// C++ program to show unary operator overloading

//#include <iostream>

//using namespace std;

//

//class Distance {

//public:

//

// // Member Object

// int feet, inch;

//

// // Constructor to initialize the object's value

// Distance(int f, int i)

// {

// this->feet = f;

// this->inch = i;

// }

//

// // Overloading(-) operator to perform decrement

// // operation of Distance object

// void operator-()

// {

// feet--;

// inch--;

// cout << "\nFeet & Inches(Decrement): " << feet << "'" << inch;

// }

//};

//

//// Driver Code

//int main()

//{

// // Declare and Initialize the constructor

// Distance d1(8, 9);

//

// // Use (-) unary operator by single operand

// -d1;

// getchar();

// getchar();

//

//}

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

// C++ program to show binary operator overloading

//

//#include <iostream>

//using namespace std;

//

//class Distance {

//public:

// // Member Object

// int feet, inch;

// // No Parameter Constructor

// Distance()

// {

// this->feet = 0;

// this->inch = 0;

// }

//

// // Constructor to initialize the object's value

// // Parametrized Constructor

// Distance(int f, int i)

// {

// this->feet = f;

// this->inch = i;

// }

//

// // Overloading (+) operator to perform addition of

// // two distance object

// Distance operator+(Distance& d2) // Call by reference

// {

// // Create an object to return

// Distance d3;

//

// // Perform addition of feet and inches

// d3.feet = this->feet + d2.feet;

// d3.inch = this->inch + d2.inch;

//

// // Return the resulting object

// return d3;

// }

//};

//

//// Driver Code

//int main()

//{

// // Declaring and Initializing first object

// Distance d1(8, 9);

//

// // Declaring and Initializing second object

// Distance d2(10, 2);

//

// // Declaring third object

// Distance d3;

//

// // Use overloaded operator

// d3 = d1 + d2;

//

// // Display the result

// cout << "\nTotal Feet & Inches: " << d3.feet << "'" << d3.inch;

// getchar();

// getchar();

//

//}

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

// C++ program to show binary operator overloading

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

//#include <iostream>

//using namespace std;

//

//class Complex

//{

//private:

// float real;

// float imag;

//public:

// Complex() : real(0), imag(0) { }

// void input()

// {

// cout << "Enter real and imaginary parts respectively: ";

// cin >> real;

// cin >> imag;

// }

//

// // Operator overloading

// Complex operator - (Complex c2)

// {

// Complex temp;

// temp.real = real - c2.real;

// temp.imag = imag - c2.imag;

//

// return temp;

// }

//

// void output()

// {

// if (imag < 0)

// cout << "Output Complex number: " << real << imag << "i";

// else

// cout << "Output Complex number: " << real << "+" << imag << "i";

// }

//};

//

//int main()

//{

// Complex c1, c2, result;

//

// cout << "Enter first complex number:\n";

// c1.input();

//

// cout << "Enter second complex number:\n";

// c2.input();

//

// // In case of operator overloading of binary operators in C++ programming,

// // the object on right hand side of operator is always assumed as argument by compiler.

// result = c1 - c2;

// result.output();

//

// getchar();

// getchar();

//}

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