//#include<iostream>

//using namespace std;

//class complex

//{

//private:

// int real, imgenary;

// public:

// complex()

// {

// real = 0;

// imgenary = 0;

// }

// complex(int x, int y)

// {

// real = x;

// imgenary = y;

//

//

// }

// void disply()

// {

// cout << "real " << real<<endl;

// cout << "imgenary" <<imgenary<<endl;

//

//

// }

// complex operator +(complex x)

// {

// complex temp;

// temp.real = real + x.real;

// temp.imgenary = imgenary + x.imgenary;

// return temp;

//

//

//

// }

//

//};

//int main()

//{

//

//

// complex c1(2, 3), c2(5, 3) , c3;

// c3=c1 + c2;

// c3.disply();

//

//

//

// system("pause");

//

//}

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

//

//#include<iostream>

//using namespace std;

//class complex

//{

//private:

// int real, imgenary;

// public:

// complex()

// {

// real = 0;

// imgenary = 0;

// }

// complex(int x, int y)

// {

// real = x;

// imgenary = y;

//

//

// }

// void disply()

// {

// cout << "real " << real<<endl;

// cout << "imgenary" <<imgenary<<endl;

//

//

// }

// int operator ==(complex x)

// {

//

// if (real ==x.real && imgenary ==x.imgenary)

//

// {

// return 1;

// }

// else

// {

// return 0;

// }

// }

//

//};

//int main()

//{

//

//

// complex c1(2, 3), c2(5, 3);

//

// if (c1==c2)

// {

// cout << "values are same";

//

// }

// else

// {

// cout << "values are not same";

//

// }

//

//

//

//

// system("pause");

//

//}

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

#include<iostream>

using namespace std;

class complex

{

private:

int real, imgenary;

public:

complex()

{

real = 0;

imgenary = 0;

}

complex(int x, int y)

{

real = x;

imgenary = y;

}

void disply()

{

cout << "real " << real<<endl;

cout << "imgenary" <<imgenary<<endl;

}

friend complex operator +(complex x,complex y)

{

complex temp;

temp.real = x.real + y.real;

temp.imgenary = x.imgenary + y.imgenary;

return temp;

}

};

int main()

{

complex c1(2, 3), c2(5, 3) , c3;

//c3= operator +(c1, c2); //by function invoke

c3 = c1 + c2; //by operator invoke

c3.disply();

system("pause");

}