Name : Mukul Kumar

Enrollment number : 201B162

Batch : B5

Subject : OOPL

Lab Assignment number : 5

Semester : 2

1. Write a program to perform addition of two complex numbers using constructor overloading. The first constructor which takes no argument is used to create objects which are not initialized, second which takes one argument is used to initialize real and imaginary parts to equal values and third which takes two argument is used to initialized real and imaginary to two different values. Include sum member function that takes two objects and performs the addition of these two objects. Write display function to display the object in complex form i.e. 2+i6 . Write main function to test your program.

**Code:**

#include <iostream>

using std::cout;

using std::endl;

class ABC

{

int real, imaginary;

public:

ABC() {}

ABC(int a)

{

real = a;

imaginary = a;

}

ABC(int a, int b)

{

real = a;

imaginary = b;

}

friend ABC sum(ABC m, ABC n)

{

ABC temp;

temp.real = m.real + n.real;

temp.imaginary = m.imaginary + n.imaginary;

return temp;

}

void display()

{

cout << real;

if (imaginary < 0)

{

cout << " - "<< -1\*imaginary <<"i"<< endl;

}

else

{

cout << " + " << imaginary <<"i"<< endl;

}

}

};

int main()

{

ABC a(2), b(2, 4), sum\_;

a.display();

b.display();

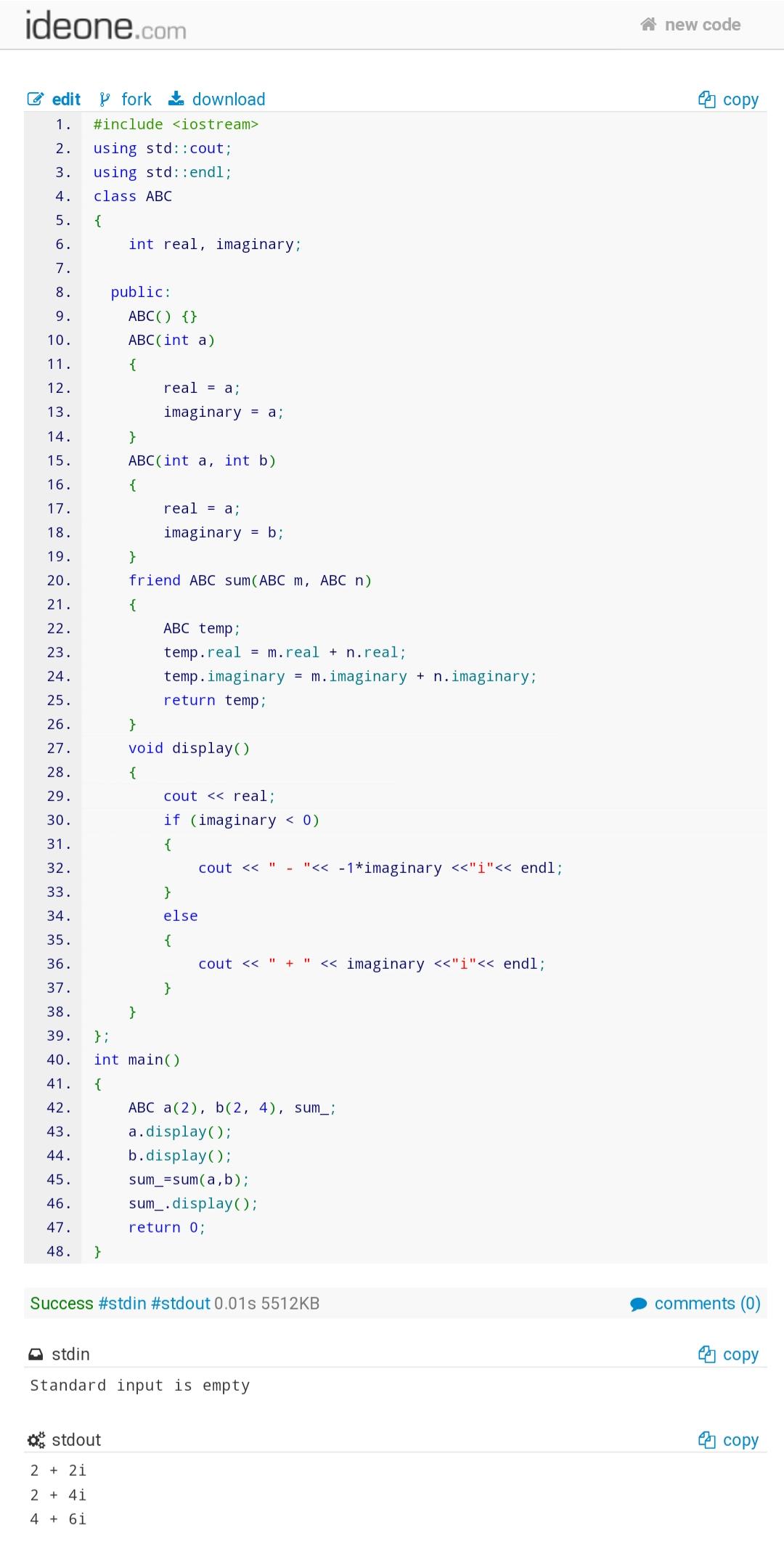
sum\_=sum(a,b);

sum\_.display();

return 0;

}

**OUTPUT:**



1. A point in a two-dimensional plane having coordinate as (x,y), can be represented by a class whose private data members are x and y. Write the constructor and destructor functions of the class. The constructor should initialise (x,y) by passing parameters values. Now, a rectangle can be represented by the top-left and bottom-right vertices. Define a class say ‘Rectangle’ whose private data members are top-left and bottom-right vertices. Write the parameterised constructor function of the class ‘Rectangle’. Also, write the destructor function. Finally, write a program to show the order in which different constructors and destructors are called.

**Code:**

#include <iostream>

using std::cout;

using std::endl;

class parameters

{

int x;

int y;

public:

parameters()

{

cout << "Default constructor of parameter class called" << endl;

}

parameters(int x, int y) : x(x), y(y)

{

cout << "Constructor of parameter class called" << endl;

}

~parameters()

{

cout << "Destructor of parameter class called" << endl;

}

parameters(const parameters &a)

{

cout << "Copy constructor of parameter class called" << endl;

}

};

class Rectangle

{

parameters top\_left;

parameters bottom\_right;

public:

Rectangle()

{

}

Rectangle(parameters top\_left, parameters bottom\_right)

{

this->top\_left = top\_left;

this->bottom\_right = bottom\_right;

cout << "Constructor of rectangle class called" << endl;

}

~Rectangle()

{

cout << "Destructor of rectangle class called" << endl;

}

};

int main()

{

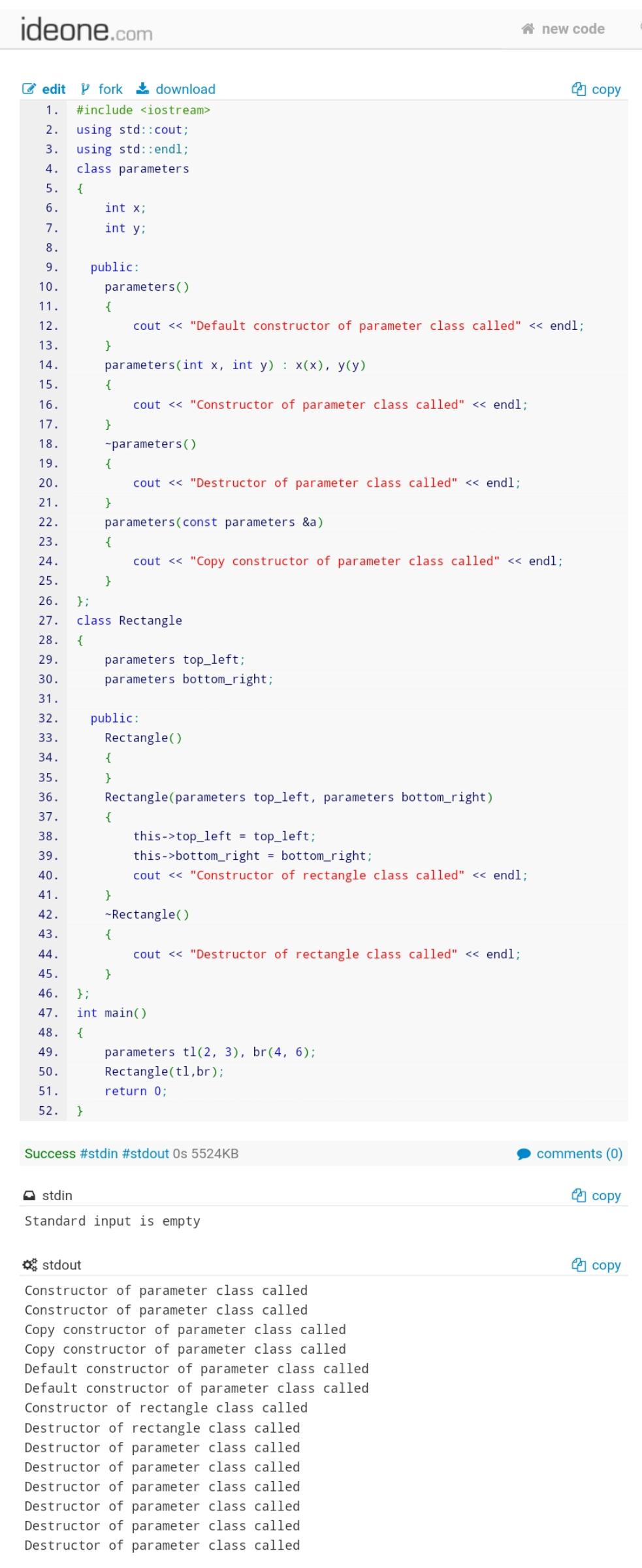
parameters tl(2, 3), br(4, 6);

Rectangle(tl,br);

return 0;

}

**OUTPUT:**



1. Write a program to show that, the constructor and destructor functions of a globally declared object are the first and last functions, respectively to be called in a program.

**Code:**

#include <iostream>

using std::cout;

using std::endl;

class ABC

{

int x;

public:

ABC(int x)

{

this->x = x;

cout << "constructor called for object having value of x : " << x << endl;

}

~ABC()

{

cout << "destructor called for object having value of x : " << x << endl;

}

};

ABC a(2);

int main()

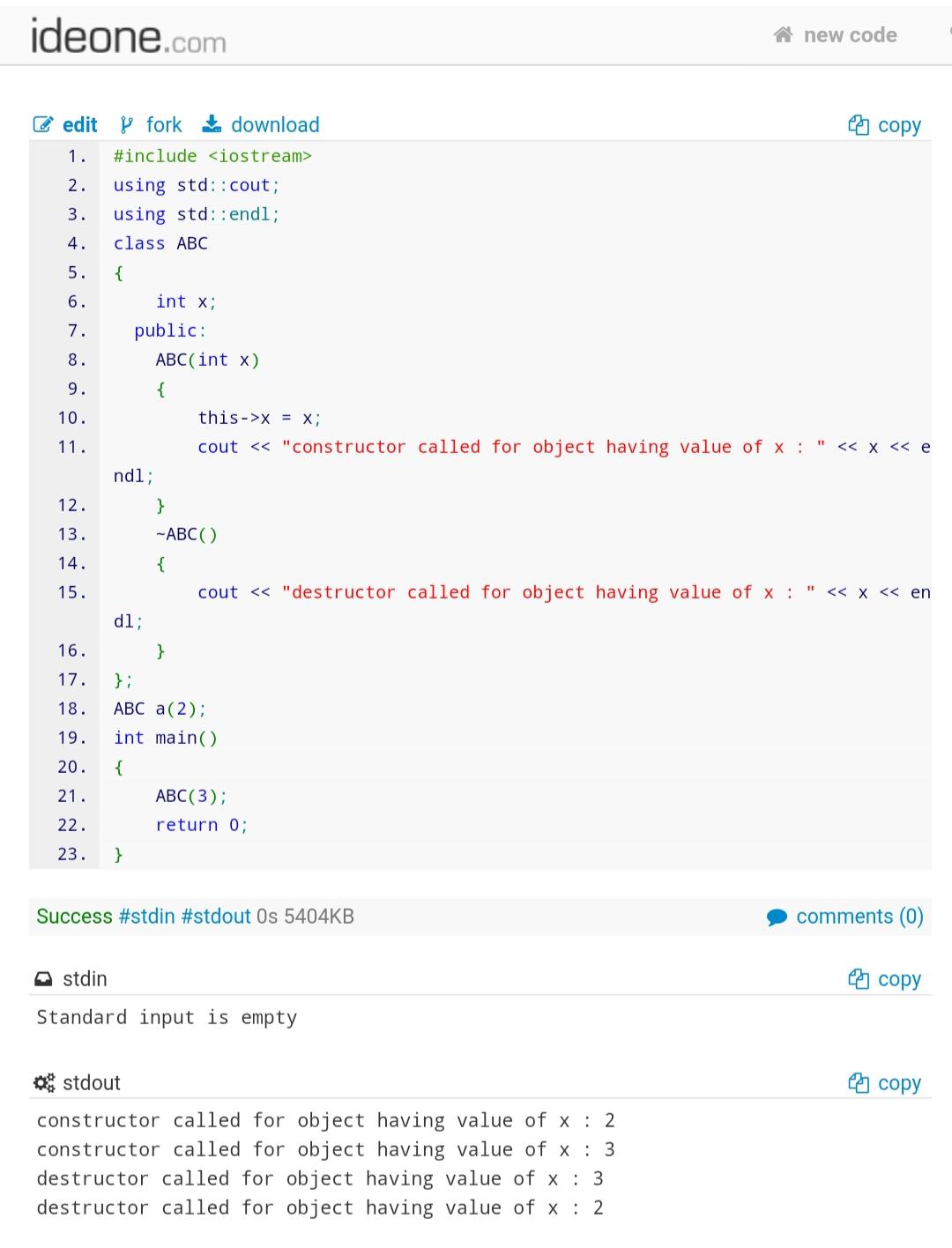
{

ABC(3);

return 0;

}

**OUTPUT:**



1. Write a program to show that constructors follow the property of function overloading as well as default parameter. Also, show that in case of constructor also, default parameter may create problem in implementing function overloading.

**Code:**

#include <iostream>

using std::cout;

using std::endl;

class add

{

private:

int num1, num2, sum;

public:

add(int n1){

num1=n1;

num2=n1;

}

add(int n1 , int n2 = 0)

{

num1 = n1;

num2 = n2;

sum = num1 + num2;

cout << "num1 + num2 =" << sum << endl;

}

};

int main()

{

add obj1(23, 45);

add obj2(5);

return 0;

}

**OUTPUT:**

