/\*\*\*\*\*\*\*\*\*\* Online on Operator Overloading \*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Time: 30 minutes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Add the required functionalities so all the operations in main() works \*/

/\* You cannot change the main() function.\*/

/\* You cannot add any getter or setter functions in the Point class\*/

#include <iostream>

using namespace std;

class Point

{

private:

int x;

int y;

public:

Point()

{

x = 0;

y = 0;

}

Point(int x, int y)

{

this->x = x;

this->y = y;

}

~Point()

{

x = 0;

y = 0;

}

friend ostream &operator<<(ostream &output, Point &p)

{

output << "(" << p.x << ", " << p.y << ")";

return output;

}

// your code

};

class Square

{

private:

Point topLeft;

int side;

public:

Square()

{

topLeft = Point(0, 0);

side = 0;

}

Square(Point topLeft, int side)

{

this->topLeft = topLeft;

this->side = side;

}

friend ostream &operator<<(ostream &output, Square &s)

{

output << "Top left point: " << s.topLeft << ", side: " << s.side;

return output;

}

// your code

};

int main()

{

Point p1(1, 2);

Point p2(3, 4);

cout << "p1: " << p1 << endl; // print p1

cout << "p2: " << p2 << endl; // print p2

++p1; // should increment the value of x coordinate by 1

cout << "++p1: (" << p1[0] << ", " << p1[1] << ")" << endl; // ++p1: (2, 2)

// p1[0] should return the value of x coordinate

// p1[1] should return the value of y coordinate

p2++; // should increment the value of y coordinate by 1

cout << "p2++: (" << p2[0] << ", " << p2[1] << ")" << endl; // p2++: (3, 5)

Point invertedP2 = ~p2; // should return a new point with inverted coordinates

// if p2 is (3, 5), invertedP2 should be (-3, -5)

cout << "Inverted p2: (" << invertedP2[0] << ", " << invertedP2[1] << ")" << endl; // Inverted p2: (-3, -5)

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

Square s1(Point(4, 5), 5);

Square s2(Point(3, 4), 10);

cout << "s1: " << s1 << endl; // print s1

cout << "s2: " << s2 << endl; // print s2

Square sumSquares = s1 + s2; // should return a new square with top left point equal to

// the sum of the top left points of s1 and s2 and

// side equal to the sum of the sides of s1 and s2

cout << "Sum of s1 and s2: top left point: " << sumSquares << endl; // Sum of s1 and s2: top left point: Top left point: (7, 9), side: 15

Square s3 = sumSquares + p2; // should return a new square with top left point equal to

// the sum of the top left point of sumSquares and p2 and

// side equal the side of sumSquares

cout << "s3: " <<s3 << endl; // s3: Top left point: (10, 14), side: 15

return 0;

}

// Expected output:

// p1: (1, 2)

// p2: (3, 4)

// ++p1: (2, 2)

// p2++: (3, 5)

// Inverted p2: (-3, -5)

// =====================

// s1: Top left point: (4, 5), side: 5

// s2: Top left point: (3, 4), side: 10

// Sum of s1 and s2: top left point: Top left point: (7, 9), side: 15

// s3: Top left point: (10, 14), side: 15