Section, Name and ID#: [redacted]

AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
CS 120 Introduction to Object-Oriented Programming
QUIZ 09

Date / Time:	Friday, April 14 2017 at 17:00
Duration:	1 hour
Attention:	**ANY TYPE OF COMMUNICATION IS STRICTLY PROHIBITED**
	Write down your section, name and ID# at the top of all used pages
Instructions:	1. Write the solutions in the top half of each page under problem statements
	2. Copy the same solution in the bottom section to take with you after quiz
	3. Turn your solution into a program, compile and submit the errors
	4. Correct the errors and submit the working version of your program
Submission Deadline:	**Sunday, April 16 2017, before 22:00**
Submission Contact:	**skhachat@aua.am**
	arshavir.voskanyan@gmail.com, nareh_salmasian@edu.aua.am

Problem 1: A rectangle with sides parallel to ***x*** and ***y*** axes can be represented by its diagonal of type ***line***. Implement a C++ class rectangle (its member functions) assuming the existence of all necessary functions of the class line:

```
class rectangle
{ public:
      rectangle(double x0, double y0, double x1, double y1); // initializes by
                                                  //bottom-left and top-right coordinates
      double perimneter();
      double area();
      bool intersect(rectangle &that); // checks if the rectangles intersect
      rectangle union(rectangle &that); // returns least rectangle that includes both
  private:
      line diagonal; // arrays of x and y coordinates of vertices respectively
}
```

```
rectangle::rectangle(double x0, double y0, double x1, double y1){
  x0 = double x0; y0 = double y0; x1 = double x1, y1 = double y1
};
double rectangle::perimneter(){
  p = 2*(diagonal.dist(x0, y0, x1, y0) + diagonal.dist(x0, y0, x0 y1)); // diagonal.distance length
    return p;
}
double rectangle::area(){
  s = diagonal.dist(x0,y0,x1, y0) * diagonal.dist(x0, y0, x0, y1);
}bool rectangle::intersect(rectangle &that){
```

Student's copy

~~Java public static int[] reverse~~

another problem, take array fill with 1, 0, 1.0....02 true false, true false

```
public static int[] reverse(int arr[]){
    bool b=true;
    for(int i=0 i<arr.length(), i++){
    arr[i] = b
      b++
}
```

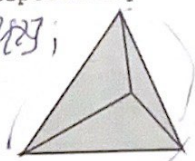
Use the backside, if needed

OOP.Q09.140411.H059

Problem 2: Implement a C++ ***class triangle*** (only its member functions marked by **TODO**) the header file of which is given below. The Heron's formula is $area = \sqrt{p(p-a)(p-b)(p-c)}$, where p is the half-perimeter and a, b and c are the sides.

```
class triangle
{ public:
     triangle(double vertex[2][3]); // TODO - initializes vertices by specified
                                    // array of two rows and three columns
     double get_x(int vertex); // returns x coordinate of specified vertex
     double get_y(int vertex); // returns y coordinate of specified vertex
     double side(int vertex); // returns side length from specified vertex to next one

     double perimneter(); // TODO
     double area(); // TODO - computes area using Heron's formula
     bool is_inside(double px, double py); // TODO - checks if a point  with coordinates
                     // (px, py) is inside the triangle - see shaded areas below
  private:
     double x[3], y[3]; // arrays of x and y coordinates of vertices respectively
}
```



```
triangle::triangle(double vertex[2][3]){ x0 = vertex[0][0]; x1 = vertex[0][1]; x2 = vertex[0][2];
                                         y0 = vertex[1][0]; y1 = v
for(int i=0; i<vertex.length; i++){
  set_x[i] = vertex[0][0];
  set_y[i] = vertex[1][0];
}

double triangle::perimeter(){
p = side(0) + side(1) + side(2)
  return p;
}
double area(){
s = sqrt((p/2)(p/2 - side(0))(p/2 - side(1))(p/2 - side(2)));
  return s;
}
bool is_inside(double px, double py){


}
```

Student's copy

Use the backside, if needed

OOP.909.140417.H059