Section, Name and ID#: OOP.009.140417.H001

OOP.009.140417.H001

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): line diagonal(
        double x0, double y0, double x1, double y1) {
    x0 = diagonal.get(x0), x1 = diagonal.get(x1);
    y0 = diagonal.get(y0); y1 = diagonal.get(y1); }
double rectangle::perimneter(){
    return 2*(y1-y0)+2*(x1-x0); }
double rectangle::area(){ return (y1-y0)*(x1-x0); }
bool rectangle::intersect(rectangle &that){
  if((diagonal.get(x1) - diagonal.get(x0) + that.diagonal.get
  (x1) - that.diagonal.get(x0)) > (max(diagonal.get(x1),
  that.diagonal.get(x1)) - min(diagonal.get(x0),
  that.diagonal.get(x0)) && the same
  for y's ) return 0; return 1; }

rectangle rectangle::union(rectangle &that){
    x0 = min(x0, that.get(x0));
    x1 = max(x1, that.get(x1));
    y0 = min(y0, that.get(y0));
    y1 = max(y1, that.get(y1)); }
```

Student's copy

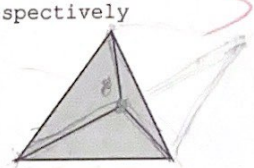
Use the backside, if needed

Section, Name and ID#: OOP.009.140417.H001

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[][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[][3]) {
x[0] = vertex[0][0];
x[1] = vertex[0][1];
x[2] = vertex[0][2];
y[0] = vertex[1][0];
y[1] = vertex[1][1];
y[2] = vertex[1][2]; }

double triangle :: perimeter(){
    int a;
    a = side(0) + side(1) + side(2);
    return a; }
```

Student's copy

```
double triangle::area() {
    return sqrt(perimeter() * (perimeter() - side(0)) *
        (perimeter() - side(1)) * (perimeter() - side(2)); }

bool triangle :: is_inside(double px, double py){
    if (area() == a.area( ) + b.area( ) +
                  c.area( ) ) return 1;
    return 0;
}

triangle a( vertex[][0] );
triangle b( vertex[][1] );
triangle c( vertex[][2] );
```

Use the backside, if needed

Section, Name and ID#: OOP.009.140417H001

Problem 3: Write and implement the following C++ classes:

1. ***class course*** – encapsulates three member variables ***string name***, ***int crtedits*** and ***double grade***.
2. ***class semester*** – encapsulates a six-element private array course subjects[6] and implements ***void set(int i, string new_name, int new_units, double new_grade)***, ***course get(int i)***, ***int total_credits()*** and ***double gpa()*** public functions. If total credits are ***0***, the gpa is also ***0***. Make appropriate changes in ***class course***.

3

```
class course {                    default const.
public: course(); course(double grade1, int credits1,
string name1);
    string name = name1;
    int credits = credits1;
    double grade = grade1; }
class semester {
  private: course subjects[6];
  public: void set(.....);
          course get(int i);
          int total_credits();
          double gpa();

void semester::set(...) {
      new_name = subject.name[i];
      new_units = subject.credit[i];
      new_grade = subject.grade[i]; }
course semester::get(int i) {
      return subject[i]; }
int semester::total_credits() { int total;
    for(int i=0; i<5; i++){
      total += subject[i].credits(); }
      return total; }
double semester::gpa() {
      int sum,
      for (int i=0; i<5; i++){
        sum += semester.new_grade[i] *
if (total_credits()==0)          semester.new_units[i]; }
  return 0;
  return sum/total.credits(); }
```

Student's copy

Use the backside, if needed

Problem 3 of 3