Section, Name and ID#:

AMERICAN UNIVERSITY OF ARMENIA

College of Science and Engineering

CS 120 Introduction to Object-Oriented Programming

MIDTERM EXAM

10

Date / Time: Friday, March 17 2017 at 17:30

Duration: 2 hours

Attention: <u>**ANY TYPE OF COMMUNICATION IS STRICTLY PROHIBITED**</u>

<u>***Write down your section, name and ID# at the top of all used pages***</u>

Participation:

Problem 1: Consider below a C++ function ***float kahan(float num1, float num2, float& compensation)*** that implements the ***Kahan Summation Algorithm*** for high-precision compensated summation of two float arguments ***float num1*** and ***float num2***:

```
float kahan(float num1, float num2, float &compensation)
{
        float result;
        num2 -= compensation;
        result = num1 + num2;
        compensation = (result - num1) - num2;
        return result;
}
```

Using this function, write a C++ function ***float e(int n)*** that computes the value ***e*** by the following formula:

$$e = \sum_{k=0}^{n} \frac{1}{k!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1*2} + \frac{1}{1*2*3} + \cdots$$

Recall that the factorial of non-positive numbers equals to ***1*** by definition.

The initial value of ***float compensation*** is ***0.0***.

function

```
float e(int n) {  float e;
for(int k=0; k<n; k++) {

~~kahan(1,~~        1/fact(k);

float  fact = ~~fact(k);~~
/ e = kahan(1, fact, 0.0);

}
```

return needed

type?

```
int res=1;  function  fact(int num) { ~~int x;~~
for(int i=0; i<num; i++)
    int res *= num;
}
```

return needed

1

Use the backside, if needed

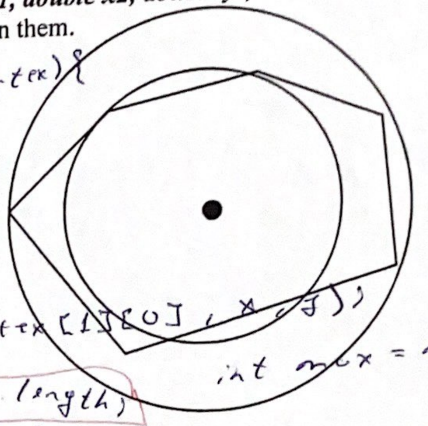
OOP.MT.170317.M015

Section, Name and ID#:

Problem 3: Write a Java function ***public static double thickness(double[][] vertex)*** that takes as its argument a ***2-by-n*** array of polygon's vertex coordinates ***double[][] vertex*** – the ***x*** coordinates in the first row and ***y*** coordinates in the second row. It returns polygon's boundary thickness as follows:

1. Computes the center – the mean ***x*** and ***y*** vertex coordinates;
2. Returns the difference between the maximal and minimal distances from the center to the vertices.

You may assume and use a method ***double dist(double x1, double y1, double x2, double y2)*** that takes as its arguments coordinates of two points and returns the distance between them.



public static double thickness (double [][] vertex){

~~int x; int y;~~ ~~int~~ double x = 0 ; ~~int~~ double y = 0;

for (int i = 0 ; i < vertex.length ; i++) {

x += vertex [0] [i];

y += vertex [1] [i];

for ~~(int j = 0 ; j < vertex~~

} int min = dist (vetex [0] [0] , vertex [1] [0] , x , y));

int max = min;

x = x / vertex .length; y = y / vertex.length)

for (int i = 0 ; i < vertex. length ; i ++) {

~~for (int j = 0 ; j < vertex [0]. length ; j ++) {~~

int ~~min~~ x = dist (vertex [0] [i] , vertx [1] [i] , x , y)

~~int max~~ if (~~min~~ ~~vertex~~ x < min)

min = x;

if (x > max)

max = x;

} return max - min;

}

! double dist (double x1, double y1, double x2, double y2) {

double dist = sqrt ((x1 - y1) (x1 - y1) + (x2 - y2)(x2 - y2));

return dist

}

3

Use the backside, if needed

OOP.MT.170317.M015