CSE102 – Computer Programming

Homework #10

Understanding Structs using Geometry

Due Date: 18/05/2020

Hand in: A student with number 20180000001 should hand in a 'c' file named 20180000001.c for this homework.

In this assignment, you are going to use structures to describe and perform computations on simple 2D geometric objects placed on Euclidean space (coordinate system). You will be using **points, lines, and polygons.** You will read the definitions of a given set of geometric structures and actions to be performed on them from an input file. The results of the actions will be printed into an ouput file.

Input File:

The input file has two parts: data and actions.

- Data:
  - Data part starts with keyword "data" (a single line).
  - It is followed by a number indicating the number of 2D objects to be read.
  - Following each line will have a 2D object definition. See below for definition of 2D objects.
  - You can assume that there will be at least 1 and at most 100 2D objects.
  - You can assume that the object names are unique (no replications).
- Actions:
  - Action part starts with keyword "actions" (a single line).
  - It is followed by a file name indicating where the results of the actions will be output.
  - This is followed by a number indicating the number of actions.
  - Following each line will have an action. See below for definition of 2D actions and what they should generate.
- Comments:
  - Any part of the input file can include a comment which should be discarded during reading. The comments starts with "\\" and ends at the end of the line.

Example input and output files are provided as attachements.

2D Objects: There are three kinds of geometric objects.

- Point: A point is defined by its two coordinates and a name. I.e.,
  - 100.0 100.0 P1 // A point at location (100,100) with name "P1".
  - 200.0 200.0 P2 // A point at location (100,100) with name "P2".
- Line: A line is defined by two points and a name. I.e.,

    - P1 P2 L12 // The line named “L12” defined by two end points “P1” and “P2P).
- Polygons: A circularly connected set of lines with at most 20 components. It can be defined either by connecting a set of points or a set of lines. I.e.,
    - P1 P2 P3 P4 PG4 // The polygon “PG4” defined by four lines connecting first “P1” and “P2”, second “P2” and “P3”, third “P3” and “P4” and finaly “P4” and “P1”.
    - L12 L23 L31 P4 PG3 // The polygon “PG3” defined by three lines “L12”, “P23” and “L31” in the given order.

Actions: The following actions can be defined over the 2D objects provided in the data part of the file.

- Distance: Print the distance between two points. For example, the following action should result in the given output.
    - Distance P1 P2 // Print out the distance between points “P1” and “P2”.
      `Distance(P1,P2) = 12.0`
- Distance: Print the distance between a point and a line. For example, the following action should result in the given output.
    - Distance P1 L12 // Print out the distance between point “P1” and line “L12”.
      `Distance(P1,L12) = 1.1`
- Angle: Print the angle (in degrees) between two lines. For example, the following action should result in the given output.
    - Angle L1 L2 // Print out the angle between lines “L1” and “L2”.
      `Angle(L1,L2) = 81.0`
- Length: Print the length of a given line. For example, the following actions should result in the given output.
    - Length L1 // Print out the length of line “L1”.
      `Length(L1) = 5.8`
- Length: Print the length (circumference) of a given polygon. For example, the following actions should result in the given output.
    - Length PG1 // Print out the circumference of the polygon “PG1”.
      `Length(PG1) = 15.2`
- Area: Print the area of a given polygon. For example, the following actions should result in the given output.
    - Area PG1 // Print out the area of the polygon “PG1”
      `Area(PG1) = 144.4`

We strongly encourage you to use structures and nested structures and arrays. While grading we will also look for the use of **pass by reference** (using as default parameter passing strategy. Your grade will also be affected by your choice of design. Finally, if you can not implement a given functionality, print NOT_IMPLEMENTED in the output file.

Part 1. <u>General Rules:</u>

1. We do not give you any function prototypes. We expect that you are experienced enough to understand when to use methods and name them. These will also be graded.
2. The program must be developed on Linux based OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won’t be tolerated.

3. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it is working partially.
4. Upload your ZIP or RAR file on to Moodle to deliver your homework. The zip file must consist of one C file that contains your solution. Name format can be found on the top of this homework sheet.
5. You can ask any question about the homework by sending an email to madede@gtu.edu.tr or by using the forum in the Moodle page of the course.