Computer Engineering Department

TED University

Lab Assignment 04
CMPE 252 C Programming, Spring 2023

Part 1 (60 points)

In this part, you are asked to complete circle_part1.c program (available in Moodle) which keeps the list of shapes in a text file. Please check the content of the example circles1.txt below.

Content of circles1.txt

```
circle 1 3 2
circle 2 3 14
circle -1 5 5
```

Each line contains a shape data. The data format for each shape type is as follows.
circle <center-x-coordinate> <center-y-coordinate> <diameter>

Follow the below steps in your program:

Create **point_t** structure with x (double) and y (double) coordinates.

Create **circle_t** structure with center (point_t) and radius (double), perimeter (double) and area (double)

Write 3 functions:

- int scanShape(FILE *filep, circle_t *objp);
  scanShape function gets a pointer to FILE and a pointer to circle_t. Reads shape data from the file and fills circle_t pointed to, by objp. Returns 1 if the read operation is successful; otherwise, returns 0.

- int loadCircles(circle_t shapes[]);
  loadCircles function gets an array of circle_t. Opens the text file with the entered name. For each array element, reads data by calling scanShape function. Stops reading when scanShape function returns 0. Returns the number of read shapes.

- void printShape(const circle_t *objp);
  printShape function gets a pointer to a constant circle_t. Prints shape information. The format for each circle is as follows (also see example run). While printing double values, use %.2lf as the format specifier. Use pi as 3.14159265358979323846 for perimeter and area calculations.
  Circle: <center-x-coordinate center-y-coordinate> <radius> <perimeter> <area>

- **main** function is already provided to you (see circle_part1.c) and it is supposed to remain as it is (you should not change it). In main function, an array of circle_t is declared, loadCircles function is called, and all circles are printed.

Example Run:

```
Enter the file name to read: circles1.txt
Opening circles1.txt
Loading complete
Closing circles1.txt

Circles:
Circle 1: <1.00 3.00> <1.00> <6.28> <3.14>
Circle 2: <2.00 3.00> <7.00> <43.96> <153.86>
Circle 3: <-1.00 5.00> <2.50> <15.70> <19.63>
```

Part 2 (40 points)

In this part, you will add the following functions to your program in Part 1.

- void shortestDistance(circle_t *objp, point_t p);
  shortestDistance function gets a pointer to circle_t and point_t p. It prints shortest distance between the given point and circle. You can use the following formula the calculate this shortest distance:
  $$\left|\sqrt{(x_1 - h)^2 + (y_1 - k)^2} - r\right|$$
  , where x_1, and y_1 is the coordinate of given points. h and k is the center x and y coordinate of given circle, respectively. Also, r is the radius of the given circle.

- **main** function is already provided to you (take main function from circle_part2.c) and it is supposed to remain as it is (you should not change it). In main function, an array of circle_t is declared, loadCircles function is called, all circles are printed. Then, shortestDistance function is called.

Example Run:

```
Enter the file name to read: circles1.txt
Opening circles1.txt
Loading complete
Closing circles1.txt

Circles:
Circle 1: <1.00 3.00> <1.00> <6.28> <3.14>
Circle 2: <2.00 3.00> <7.00> <43.96> <153.86>
Circle 3: <-1.00 5.00> <2.50> <15.70> <19.63>

Enter the point coordinate : 0 0

Shortest distances are:
Circle 1: 2.16
Circle 2: 3.39
Circle 3: 2.60
```