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 square_part1.c program (available in Moodle) which keeps the list of shapes in a text file. Please check the content of the example shapes1.txt below.

Content of squares1.txt
square 4 -5 3
square 3 -2 1
square -4 -1 5

Each line contains a shape data. The data format for each shape type is as follows:
square <center-x-coordinate> <center-y-coordinate> <side-length>

Follow the below steps in your program:

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

Create **square_t** structure with bottom left corner (point_t), bottom right corner (point_t), upper left corner (point_t), upper right corner (point_t) -, and side (double).

Write 3 functions:

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

- int loadShapes(square_t shapes[]);
  loadSquares function gets an array of square_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 square_t *objp);
  printShape function gets a pointer to a constant square_t. Prints shape information. The format for each shape type is as follows (also see example run). While printing double values, use %.2lf as the format specifier.
  Square: <bottom-left-corner-x-coordinate bottom-left-corner-y-coordinate> <bottom-right-corner-x-coordinate bottom-right-corner-y-coordinate> <upper-left-corner-x-coordinate upper-left-corner-y-coordinate> <upper-right-corner-x-coordinate upper-right-corner-y-coordinate> <side-length>

- **main** function is already provided to you (see square_part1.c) and it is supposed to remain as it is (you should not change it). In main function, an array of square_t is declared, loadShapes function is called, and all squares are printed.

Example Run:

```
Enter the file name to read: squares1.txt
Opening squares1.txt
Loading complete
Closing squares1.txt

Squares:
Square 0: <2.50 -6.50> <5.50 -6.50> <2.50 -3.50> <5.50 -3.50> <3.00>
Square 1: <2.50 -2.50> <3.50 -2.50> <2.50 -1.50> <3.50 -1.50> <1.00>
Square 2: <-6.50 -3.50> <-1.50 -3.50> <-6.50 1.50> <-1.50 1.50> <5.00>
```

Part 2 (40 points)

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

- void centerDistance(const point_t *ptp, const square_t *objp);
  centerDistance function gets a pointer to a constant point_t and a pointer to a constant square_t. prints point distance to square center. You can use Euclidian distance formula to calculate the distance between center of square and given point.

  Euclidean distance between two points (x1, y1) and (x2, y2) can be computed as:

  $$\sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

- **main** function is already provided to you (take main function from square_part2.c) and it is supposed to remain as it is (you should not change it). In main function, an array of square_t is declared, loadSquares function is called, all squares are printed, x and y coordinates of the user entered point are read and finally, distance of the user entered point from the center of each square is printed.

Example Run:

```
Enter the file name to read: squares1.txt
Opening squares1.txt
Loading complete
Closing squares1.txt

Squares:
Square 0: <2.50 -6.50> <5.50 -6.50> <2.50 -3.50> <5.50 -3.50> <3.00>
Square 1: <2.50 -2.50> <3.50 -2.50> <2.50 -1.50> <3.50 -1.50> <1.00>
Square 2: <-6.50 -3.50> <-1.50 -3.50> <-6.50 1.50> <-1.50 1.50> <5.00>

Enter x and y coordinate of the point: 0 0

Center distances are:
Square 1: 6.40
Square 2: 3.61
Square 3: 4.12
```