Computer Engineering Department

TED University

Lab Assignment 04
CMPE 252 C Programming, Spring 2023

Part 1 (70 points)

In this part, you are asked to complete rectangle_part1.c program (available in Moodle) which keeps the list of shapes in a text file. Please check the content of the example rectangles1.txt below.

Content of rectangles1.txt
rectangle -3 4 4 5
rectangle -3 -4 2 5
rectangle 3 4 4 5

Each line contains a rectangle data. The data format for each shape type is as follows:
rectangle <center-x-coordinate> <center-y-coordinate> <width> <height>

Follow the below steps in your program:

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

Create **rectangle_t** structure with bottom left corner (point_t), width (double), height (double) and perimeter (double).

Write 3 functions:

- int scanShape(FILE *filep, rectangle_t *objp);
  scanShape function gets a pointer to FILE and a pointer to rectangle_t. Reads shape data from the file, fills rectangle_t pointed to, by objp, and computes the perimeter of the shape. Returns 1 if the read operation is successful; otherwise, returns 0.
  - The perimeter is equal to 2*(width +height).

- int loadShapes(rectangle_t shapes[]);
  loadRectangle function gets an array of rectangle_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 rectangle_t *objp);
  printShape function gets a pointer to a constant rectangle_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.
  Rectangle: <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>– <width> <height> P <perimeter>

- **main** function is already provided to you (see rectangle_part1.c) and it is supposed to remain as it is (you should not change it). In main function, an array of rectangle_t is declared, loadRectangle function is called, and all rectangles are printed.

Example Run:

```
Enter the file name to read: rectangles1.txt
Opening rectangles1.txt
Loading complete
Closing rectangles1.txt

Rectangles:
Rectangle 1: <-5.00 1.50> <-1.00 1.50> <-5.00 6.50> <-1.00 6.50> P <18.00>
Rectangle 2: <-4.00 -6.50> <-2.00 -6.50> <-4.00 -1.50> <-2.00 -1.50> P <14.00>
Rectangle 3: <1.00 1.50> <5.00 1.50> <1.00 6.50> <5.00 6.50> P <18.00>
```

Part 2 (30 points)

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

- int isPerimeterBetween(const rectangle_t *objp, double minPerimeter, double maxPerimeter); **isPerimeterBetween** function gets a pointer to a constant rectangle_t, a double minPerimeter and a double maxPerimeter. Returns 1 if the perimeter of the rectange is between minPerimeter and maxPerimeter; otherwise, returns 0.

- **main** function is already provided to you (take main function from rectangle_part2.c) and it is supposed to remain as it is (you should not change it). In main function, an array of rectangle_t is declared, loadRectangles function is called, all rectangles are printed, and finally, only the rectangles whose perimeter is between user inputs are printed.

Example Run:

```
Enter the file name to read: rectangles1.txt
Opening rectangles1.txt
Loading complete
Closing rectangles1.txt

Rectangles:
Rectangle 1: <-5.00 1.50> <-1.00 1.50> <-5.00 6.50> <-1.00 6.50> P <18.00>
Rectangle 2: <-4.00 -6.50> <-2.00 -6.50> <-4.00 -1.50> <-2.00 -1.50> P <14.00>
Rectangle 3: <1.00 1.50> <5.00 1.50> <1.00 6.50> <5.00 6.50> P <18.00>

Enter minimum perimeter: 15
Enter maximum perimeter: 20

The following rectangles satisfy user conditions:
Rectangle 1: <-5.00 1.50> <-1.00 1.50> <-5.00 6.50> <-1.00 6.50> P <18.00>
Rectangle 3: <1.00 1.50> <5.00 1.50> <1.00 6.50> <5.00 6.50> P <18.00>
```