INDIAN STATISTICAL INSTITUTE

Laboratory Test IV

M. Tech (CS) - I Year, 2015-2016 (Semester - I)

Data and File Structures Laboratory

Date: 07.11.2015 Total Marks: 180 + 20 = 200 (20 marks for good programming habits)

Note: Follow the file naming convention strictly as mentioned.

You are not allowed to connect and browse the internet during the test. No books and e-books are allowed. You can reuse your own code.

Any instance of malpractice would be dealt with sternly. If you are in doubt about whether your action is improper, better clarify.

(Q1) Consider a railway route with n stations, numbered 1 to n, and a single train, with m berths, numbered 1 to m, running on that route from station 1 to station n. We want to design an event driven priority-based reservation system with functionalities associated with each event. The events are

1 insert a passenger;

2 book berths for passengers;

3 display passenger details.

The functionalities associated with the above events are as follows:

(1: Insert a passenger:) A passenger is added with a priority to an already existing set S of other un-allotted passengers. For a passenger, the system will record the name, sex, age, and the source and destination stations. Priority among passengers are determined as follows – a female passenger has a higher priority than a male passenger, the passenger with higher age gets priority if the sex of two passengers are the same, and first-com-first-serve is opted if both sex and age are the same. Each passenger should be allotted a unique ID – the PNR.

(2: Book berths for passengers:) The passengers in the set S are allotted berths according to their priorities, determined by their sex and age, based on the following scheme. Passengers are considered from S according to their priorities. If a passenger wants to travel between station i and station j (with $n \geq j > i \geq 1$) and if a berth is available on the train for journey between those two cities, then the berth is allotted; else, the passenger remains in the set S.

(3: Display passenger details:) Display the detailed list (PNR, name, sex and age of passenger, source and destination stations) of passengers with allotted berths, as well as the detailed list of passengers still awaiting reservation, that is, the passengers in set S.

Develop a program, preferably in C, to implement the above reservation system. Write a few lines about your data-structural design in a commented section at the beginning of your program.

Total marks: [20 + 30 + 20 = 70]

(Q2) Let A and B be two convex polygons of n and m vertices, respectively. Polygons A and B will be provided by the user as a list of vertices, not in any particular order. Develop a program to determine whether B lies completely inside A. Assume no degeneracies – i.e., A and B do not share any vertices and no three points among the vertices of A and B are collinear. See Figure 1.

Total marks: [40]

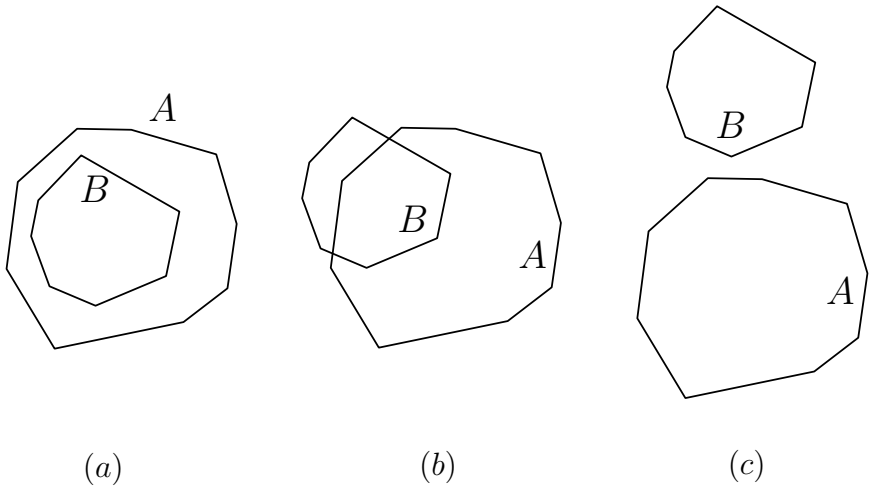

Figure 1: There are three different cases for the polygons – (a) B lies completely inside A; (b) B and A intersect; (c) B and A do not intersect. You will *only* have to identify if Case (a) is true.

(Q3) Recall the *Snakes and Ladders* game – there is a 10×10 board, with the cells numbered from 1 to 100, in the regular order as shown in Figure 2. Each cell is one of the following three types.

Blank – nothing special happens when the player lands up on this cell; 50 of the hundred cells in total, including the two cells numbered 1 and 100, are blank.

Ladder – takes the player to some cell of a higher number; 25 of the remaining cells have ladders.

Snake – takes the player down to some cell with a lower number; remaining 25 cells have snakes.

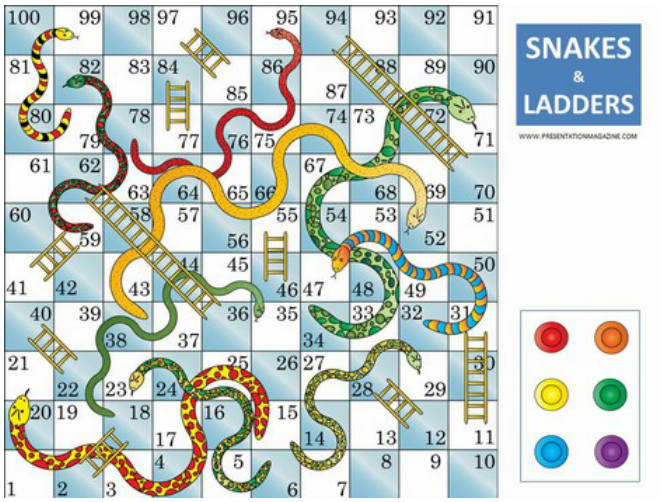

Figure 2: Sample board for the *Snakes and Ladders* game, depicting six players.

Board: The amount of jump by a ladder or the amount of fall through a snake is not specified. You may decide the values of the jumps randomly when you create your own board. But note that the player must land up on some cell within the board, i.e., within the numbers 1 to 100.

To generate various random numbers for the game – for creating the board and to roll the die – you may use the `Math.random()` function, which returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0.

Rules: At each round, each player rolls a regular die (six faces) and makes a move accordingly. A player needs to first roll a 1 on the die to get started; otherwise (s)he remains stuck at zero, i.e., outside the board. The game finishes once any one player reaches the cell numbered 100 (note that you can not land at any number greater than 100 by a jump, as described above).

Task: Design and implement the Java classes for the backend of the above game (you may call it `SNLGame`). Make the design as object-oriented as you can. You will have to write the `SNLGame`, `Player` and `Die` classes, and more classes appropriately based on how you make your design.

The final test class with a main method should act as a player and should perform the actions shown in the following code snippet.

```
// First, create a board and save it in a file
// (this should be a public static method)
SNLGame.writeBoard("board_file_name");

// Now, or later on, or even tomorrow,
// initialize an SNLGame with a board saved in a file
SNLGame game = new SNLGame("board_file_name");

// Initialize players from a given number of players
Player[] players = new Player[number_of_players];

// Also initialize an unbiased die
Die die = new Die();

// Now the game starts
boolean finished = false;
Player winner = null;
int round = 0;

// At each round of moves
while (!finished) {
    for (int i = 0; i < players.length; i++) {
        // Roll the die
        int roll = die.roll();

        // Submit the player object and the number obtained
        // in the roll to the game.
        // The game should set the new position of the player
        game.makeMove(players[i], roll);
```

```
            // If the player has reached exactly 100, finish!
            if (players[i].position() == 100) {
                finished = true;
                winner = players[i];
            }
        }
        // Optionally, print the positions of the players
    }
    // Declare the winner, print the player's Id or name.
```

Total marks: [30 + 10 + 10 + 20 = 70]

Submission instruction: Submit all your source codes to the directory `labtest4` under the home directory of the user `pdslab`. For example, if you are copying a file `labtest4-prob1-cs15XX.c`, you should use the command

```
cp labtest4-prob1-cs15XX.c ~pdslab/labtest4/
```