COP-2210 – Project

Objective

Students will be able to develop Java programs involving elements of the Java language studied along the course, including operators, input/output, decision statements, loops, arrays and classes.

Guidelines

- The programming assignment is to be completed in teams of 2 – 3 students
- The team should designate a single person (the "team leader") to submit the project deliverables.
- The assignment should be completed and submitted by Friday, March 22, 11:59 pm. Late work is not accepted except for documented emergencies.
- To submit, the team leader will upload the project solution (see *Deliverables* below) to the dropbox in Canvas.
- The first lines of your code must contain a comment with the following format:

```
/**
    Team leader: <name>, <PID>, <class section>
    Student: <name>, <PID>, <class section>

    Exercise description
*/
```

  For example:

```
/**
    Team leader: Joan Smith
    Student: Dean Green

    Exercise 1 of section Programming Projects
       in Chapter 1, page 1 of our textbook.
*/
```

Deliverables

• A compressed folder named “Project”. The folder will contain the source code file (the .java file), as well as screenshots of the running program.

Problem Description

From the following list of problems, select **one** and develop a Java project that implements a solution.

1 - Exercise 7.1 of section Programming Projects in Chapter 7, page 366 of our textbook:

P7.1 A *run* is a sequence of adjacent repeated values. Write a program that generates a sequence of 20 random die tosses in an array and that prints the die values, marking the runs by including them in parentheses, like this:

```
1 2 (5 5) 3 1 2 4 3 (2 2 2 2) 3 6 (5 5) 6 3 1
```

Use the following pseudocode:

```
Set a boolean variable inRun to false.
For each valid index i in the array
  If inRun
    If values[i] is different from the preceding value
      Print ).
      inRun = false.
  If not inRun
    If values[i] is the same as the following value
      Print (.
      inRun = true.
  Print values[i].
If inRun, print ).
```

2 - Exercise 7.2 of section Programming Projects in Chapter 7, page 366 of our textbook:

P7.2 Write a program that generates a sequence of 20 random die tosses in an array and that prints the die values, marking only the longest run, like this:

```
1 2 5 5 3 1 2 4 3 (2 2 2 2) 3 6 5 5 6 3 1
```

If there is more than one run of maximum length, mark the first one.

3 - Exercise 7.3 of section Programming Projects in Chapter 7, page 366 of our textbook.

P7.3 It is a well-researched fact that men in a restroom generally prefer to maximize their distance from already occupied stalls, by occupying the middle of the longest sequence of unoccupied places.

For example, consider the situation where ten stalls are empty.

```
_ _ _ _ _ _ _ _ _ _
```

The first visitor will occupy a middle position:

```
_ _ _ _ _ X _ _ _ _
```

The next visitor will be in the middle of the empty area at the left.

```
_ _ X _ _ X _ _ _ _
```

Write a program that reads the number of stalls and then prints out diagrams in the format given above when the stalls become filled, one at a time. *Hint:* Use an array of `boolean` values to indicate whether a stall is occupied.

4 - Exercise 7.7 of section Programming Projects in Chapter 7, page 366 of our textbook.

P7.7 A theater seating chart is implemented as a two-dimensional array of ticket prices, like this:

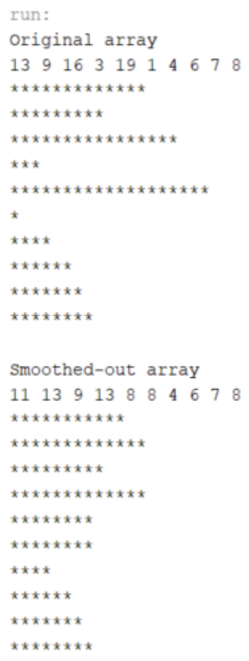
```
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10
10 10 20 20 20 20 20 20 10 10
10 10 20 20 20 20 20 20 10 10
10 10 20 20 20 20 20 20 10 10
20 20 30 30 40 40 30 30 20 20
20 30 30 40 50 50 40 30 30 20
30 40 50 50 50 50 50 50 40 30
```

Write a program that prompts users to pick either a seat or a price. Mark sold seats by changing the price to 0. When a user specifies a seat, make sure it is available. When a user specifies a price, find any seat with that price.

5 – (A variation of Exercise 7.16 of section Programming Projects in Chapter 7, page 366 of our textbook).

Sample values from an experiment often need to be smoothed out. One simple approach is to replace each value in an array with the average of the value and its two neighboring values (or one neighboring value if it is at either end of the array). Given an integer array values, smooth it out and print it as illustrated in the example below. Populate the array with random numbers. You should not create another array in your solution.

Example of the execution of such a program:



```
run:
Original array
13 9 16 3 19 1 4 6 7 8
*************
*********
****************
***
*******************
*
****
******
*******
********

Smoothed-out array
11 13 9 13 8 8 4 6 7 8
***********
*************
*********
*************
********
********
****
******
*******
********
```

6 - Write a program that simulates a statistics tool, by letting the user determine the *minimum*, the *maximum*, the *range* (maximum – minimum), the *mode* (value that is repeated more often), the *mean* (average), and the *standard deviation* of a set of numbers. These numbers, that will be stored in an array, represent salaries in the $30K – $60K range and are to be generated randomly.

This is the definition of standard deviation,

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}$$

where **x̄** denotes the average, and **n** is the total number of elements in the array.

The program will present the user a menu with these choices:

- Populate Array
- Display Array
- Minimum
- Maximum
- Range

- Mode
- Mean
- Standard Deviation
- Exit

and will let the user run them until “Exit” is entered.

Grading Rubric

The project is worth 110 points (out of 1000 total course points). The following aspects will be graded:

Aspect	**Points**	**Description**
Submission	5	The team has submitted the project solution using the requirements for deliverables specified in the *Deliverables* section.
Organization	5	Code is expected to be neat, organized, and readable.
Content	100	The code successfully implements a solution for the chosen problem.