ASSIGNMENT 3

COMP-202, Fall 2017, All Sections

Due: Wednesday, November 8^{th}, 11:59pm

Please read the entire PDF before starting. You must do this assignment individually.

Question 1:	100 points
	100 points total

It is very important that you follow the directions as closely as possible. The directions, while perhaps tedious, are designed to make it as easy as possible for the TAs to mark the assignments by letting them run your assignment, in some cases through automated tests. While these tests will never be used to determine your entire grade, they speed up the process significantly, which allows the TAs to provide better feedback and not waste time on administrative details. Plus, if the TA is in a good mood while he or she is grading, then that increases the chance of them giving out partial marks. :)

Up to 30% can be removed for bad indentation of your code as well as omitting comments, coding structure, or missing files. Marks will be removed as well if the class and method names are not respected. Make sure that you match the capitalisation of method and class names.

To get full marks, you must:

- Follow all directions below
- Make sure that your code compiles
  - Non-compiling code will receive a very low mark
- Write your name and student name is written as a comment in all .java files you hand in
- Indent your code properly
- Name your variables appropriately
  - The purpose of each variable should be obvious from the name
- Comment your work
  - A comment every line is not needed, but there should be enough comments to fully understand your program

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then we suggest that you consult the TAs during their office hours; they can help you and work with you through the warm-up questions. You are responsible for knowing all of the material in these questions.*

Warm-up Question 1 (0 points)

This program prints the outline of a square made up of * signs. It should take as input the length of the sides in number of *'s. This program should use only two *for loops*, and use *if statements* within the *for loops* to decide whether to draw a space or a star. Draw the outline of a square as follows.

```
**********
*        *
*        *
*        *
*        *
*        *
*        *
*        *
*        *
**********
```

N.B. It is normal that the square does not appear to be a perfect square on screen as the width and the length of the characters are not equal.

How do you generalize the program in order to print a rectangle where both the base and the height are given as input?

Warm-up Question 2 (0 points)

Write a program to display the (x,y) coordinates up to (9,9) of the upper right quadrant of a Cartesian plane. As in the previous warm-up question, your solution should use two nested *for loops*. Your program should also display the axes, by checking to see if the x-coordinate is zero or if the y-coordinate is zero. Note that when both the x and y coordinates are zero, you should print a + character. For example, the output of your code should look like:

```
^
|(1,9)(2,9)(3,9)(4,9)(5,9)(6,9)(7,9)(8,9)(9,9)
|(1,8)(2,8)(3,8)(4,8)(5,8)(6,8)(7,8)(8,8)(9,8)
|(1,7)(2,7)(3,7)(4,7)(5,7)(6,7)(7,7)(8,7)(9,7)
|(1,6)(2,6)(3,6)(4,6)(5,6)(6,6)(7,6)(8,6)(9,6)
|(1,5)(2,5)(3,5)(4,5)(5,5)(6,5)(7,5)(8,5)(9,5)
|(1,4)(2,4)(3,4)(4,4)(5,4)(6,4)(7,4)(8,4)(9,4)
|(1,3)(2,3)(3,3)(4,3)(5,3)(6,3)(7,3)(8,3)(9,3)
|(1,2)(2,2)(3,2)(4,2)(5,2)(6,2)(7,2)(8,2)(9,2)
|(1,1)(2,1)(3,1)(4,1)(5,1)(6,1)(7,1)(8,1)(9,1)
+---------------------------------------------->
```

Note that in the above image, all of the coordinates containing 0's are not displayed, since we are printing axes instead.

Warm-up Question 3 (0 points)

Write a method `longestSubArray` that takes as input an array of integer arrays (i.e. a multi-dimensional array) and returns the *length* of the longest sub-array. For example, if the input is: `int[][] arr= {{1,2,1},{8,6}};` then `longestSubArray` should return 3.

Warm-up Question 4 (0 points)

Write a method `subArraySame` that takes as input an array of integer arrays (i.e. a multi-dimensional

array) and checks if all of the numbers in each 'sub-array' are the same. For example, if the input is: `int[][] arr= {{1,1,1},{6,6}};` then `subArraySame` should return true and if the input is: `int[][] arr= {{1,6,1},{6,6}};` then `subArraySame` should return false.

Warm-up Question 5 (0 points)

Write a method `largestAverage` that takes as input an array of double arrays (i.e. a multi-dimensional array) and returns the double array with the largest *average* value. For example, if the input is: `double[][] arr= {{1.5,2.3,5.7},{12.5,-50.25}};` then `largestAverage` should return the array $\{1.5, 2.3, 5.7\}$ (as the average value is 3.17).

Part 2

The questions in this part of the assignment will be graded.

Question 1: Tic Tac Toe (100 points)

Before starting this question, we strongly recommend that you complete the warm-up exercises if you have not already done so. These warm-up questions will be the best way to start the assignment.

This assignment should be solved without the use of OOP concepts. In particular, all the methods must be static, and all the variables must be local.

For this question, you will write a program called `TicTacToe` that allows a user to play Tic Tac Toe against an AI (the computer). You can assume that the user plays using the symbol `'x'`, while the AI uses the symbol `'o'` (the lower case vowel, please do not use a zero or the upper case vowel instead).

To do so, write (at a minimum) the following methods. You can write any additional helper method if you want to.

A. A method called `createBoard` that takes as input one integer `n`, representing the dimension of the board, and returns an `n` by `n` array of characters. This 2 dimensional array of characters represents the board of the game. When the board is created, it should be completely empty. To represent this, the elements of the array should all be initialized with the space character `' '`. For example, `createBoard(3)` should return a reference to the following array:
`{{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}}`.

B. A method called `displayBoard` that takes a 2 dimensional array of character as input and prints out the board. Below is an example of how an empty 4 by 4 board might look like when printed out.

```
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
| | | | |
+-+-+-+-+
```

Consider the array of character `a = {{'x', ' ', ' '}, {' ', 'o', ' '}, {'x', 'o', ' '}}`, then `displayBoard(a)` should print:

```
+-+-+-+
|x| | |
+-+-+-+
| |o| |
+-+-+-+
|x|o| |
+-+-+-+
```

C. A method called `writeOnBoard` that takes as input the board (a 2 dimensional array of character), the character to write, and two integers `x` and `y` representing the position on the board where the character should be written on. Assume that the first integer indicates the row, and the second integer indicates the column. Then, a cell on a board of dimension n is represented by the coordinates `(x,y)` where `x` and `y` are integers between 0 and $n-1$. The method should first verify that the inputs received are valid: if the coordinates received represent a cell outside of the board or if the cell already contain a character that is not the space character, then throw an

IllegalArgumentException. Depending on why the exception was thrown, the appropriate error message should be printed. If the input provided to the method is valid, then the method should add the character received as input on the board in position `(x,y)`. Note that this method must be `void`.

Consider the following array of characters:

```
a = {{'x', ' ', ' '}, {' ', 'o', ' '}, {'x', 'o', ' '}}
```

Then:

- the method call `writeOnBoard(a,'x',1,1)` should cause a run-time error because the cell in position `(1,1)` contains a character other than the space character.
- the method call `writeOnBoard(a,'x',1,5)` should cause a run-time error because the position `(1,5)` does not represent a cell on the board.
- after the method call `writeOnBoard(a,'x',1,0)` the array variable `a` will be referring the following array: {{'x', ' ', ' '}, {'x', 'o', ' '}, {'x', 'o', ' '}}

D. A method called `getUserMove` that takes the board as input and returns no value. This method uses Scanner to get a move from the user. A move is composed by two integers representing the position on the board where the user wants to write their symbol (`'x'`). As before, assume that the first integer indicates the row, and the second integer indicates the column. If the move is invalid (such cell does not exist on the board, or it is already occupied by an `'x'` or an `'o'`), then ask the user to enter a new move. Keep asking the user for a new move, until they enter a valid one. Once the method receives a valid move, it carries it out by calling the `writeOnBoard` method with the appropriate inputs.

E. A method called `checkForObviousMove` that takes the board as input and returns `true` if there's an "obvious move" the AI should do, `false` otherwise. We consider to be an "obvious move" a move that would make the AI either win or avoid an obvious win for the user on the next turn. If such a move exist, then the method should carry it out by calling the `writeOnBoard` method with the appropriate inputs, and then return `true`. Note that the AI should chose to win the game over blocking an obvious win for the user. Remember also that the AI always uses the symbol `'o'`, while the user uses `'x'`. If no obvious move is possible, then the method should simply return `false`.

For example, consider the following array of characters:

```
a = {{'x', ' ', ' '}, {' ', 'o', ' '}, {'x', 'o', ' '}}
```

Then the obvious move for the AI is to place its mark (`'o'`) in position `(0,1)` since this would make the AI win the game. Therefore, if the method `checkForObviousMove` is called with input `a`, then after the call is executed, the array will be equal to {{'x', 'o', ' '}, {'', 'o', ' '}, {'x', 'o', ' '}} and the method will return `true`.

F. A method called `getAIMove` that takes the board as input and returns no value. This method should first check whether an "obvious move" is possible for the AI and carry it out by calling the `checkForObviousMove` method. If no "obvious move" was possible, then the method uses the Random class to generate a move for the AI. If the move generated is invalid (the cell is already occupied by a character that is not a space), then the method generates a new one. Once the method has generated a valid move, then it carries it out by calling the `writeOnBoard` method with the appropriate inputs.

G. A method called `checkForWinner` that takes the board as input and returns a character. The method should check whether either the user or the AI have won the game. If the user won the game, then the method returns `'x'`, if the AI won the game then it returns `'o'`, if there is no

winner yet it returns `' '` (the space character!). Remember that a player wins the game if its mark appears in every cell of a row, column, or diagonal.

For example, consider the following arrays of characters:

```
a = {{'x', 'o', ' '}, {' ', 'o', ' '}, {'x', 'o', ' '}}
b = {{'x', ' ', ' '}, {' ', 'x', ' '}, {' ', 'o', 'x'}}
c = {{'x', 'o', ' '}, {' ', 'o', ' '}, {' ', ' ', 'x'}}
```

Then `checkForWinner` with input `a` returns `'o'`, with input `b` returns `'x'`, and with input `c` returns `' '`.

H. A method called `play` that takes no inputs and returns no value. This method should implement a game of Tic Tac Toe between the user and the AI using all the methods previously defined. The method uses Scanner to take inputs from the user. Before beginning the actual game, the method should ask the user for their name and store it in an appropriate variable. Then, the method should ask the user for an integer indicating the dimension of the board the user wants to play with. If the user does not input an integer, then the method should keep asking for an input of the correct type until it receives one. The method can then start to carry out a game of Tic Tac Toe. It first creates a board with the correct dimension, then flip a coin to decide whether the user or the AI should start to play. The method should print out the result of the coin toss so that the user knows who has the first move. After the order of the players has been decided, the players alternate each other in making a move. The method should display the updated board each time a move is made. The players keep taking turns until either one of them wins or there are no more available moves (note that both players together have a maximum of $n \times n$ moves, where n is the dimension of the board). The method should then print a message on the screen displaying the result of the game.

   On the next few pages you can find an example of a user (Cersei) playing one game.

The following are some examples of useful helper methods that you might want to create in order to make your program easier to read, code, and debug:

- A method to check whether the AI is winning (or losing) on a row. A method to check whether the AI is winning (or losing) on a column. A method to check whether the AI is winning (or losing) on a diagonal. These methods will help you build the method that should check for an obvious move.
- A method that looks for a cell with a space character along a row/column/diagonal (i.e. given a 1 dimensional array of characters).
- A method that takes a 1 dimensional array of characters and a char `c` as input. The method counts how many `c`'s are in the array.
- A method that takes the board as input and returns a column of the board (as a 1 dimensional array of characters)
- A method that takes the board as input and returns one of the two diagonals (as a 1 dimensional array of characters)
- A method that prints a line of characters alternating between `'+'` and `'-'`.

The above are just some examples. Feel free to write as many helper methods as you'd like. Your assignment will be graded based on whether the methods from A to H work correctly.

```
Please enter your name:
Cersei
Welcome, Cersei! Are you ready to play?
Please chose the dimension of your board: 3
The result of the coint toss is: 1
The AI has the first move
The AI made its move:
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
| | |o|
+-+-+-+
Please enter your move: 0 1
+-+-+-+
| |x| |
+-+-+-+
| | | |
+-+-+-+
| | |o|
+-+-+-+
The AI made its move:
+-+-+-+
| |x| |
+-+-+-+
|o| | |
+-+-+-+
| | |o|
+-+-+-+
```

```
Please enter your move: 02
+-+-+-+
| |x|x|
+-+-+-+
|o| | |
+-+-+-+
| | |o|
+-+-+-+
The AI made its move:
+-+-+-+
|o|x|x|
+-+-+-+
|o| | |
+-+-+-+
| | |o|
+-+-+-+
Please enter your move: 20
+-+-+-+
|o|x|x|
+-+-+-+
|o| | |
+-+-+-+
|x| |o|
+-+-+-+
The AI made its move:
+-+-+-+
|o|x|x|
+-+-+-+
|o|o| |
+-+-+-+
|x| |o|
+-+-+-+

GAME OVER!
You lost
```

What To Submit

Please put all your files in a folder called Assignment3_ID, where 'ID' should be replaced by your McGill ID number. Zip the folder (please DO NOT rar it) and submit it in MyCourses. Inside your zipped folder there must be the files listed below. **Do not submit any other files, especially .class files**.

`TicTacToe.java`

`Confession.txt` (optional) In this file, you can tell the TA about any issues you ran into doing this assignment. If you point out an error that you know occurs in your problem, it may lead the TA to give you more partial credit. On the other hand, it also may lead the TA to notice something that otherwise they would not.