COS 126 General Computer Science Spring 2022 Programming Exam 1

Before the exam. Read this page of instructions before the exam begins. However, do not start the exam (not even reading the next page!) until instructed to begin.

Exam duration. Once the exam begins, you have eighty (80) minutes to complete it.

Submission & Grading. As with the COS 126 assignments, you can and should submit your code multiple times, but only the last version will be graded. You are responsible for uploading the correct version of your solution. Your final submitted program *must* compile; if not, expect your grade to be 0. Your program will be graded mainly on correctness; but clarity and design may also be considered.

Resources. This exam is "open-book" but not "open-internet" – for example do not use Google to find the answer to questions like: "How to do X in Java." During the exam you may only use: the textbook, the booksite, your notes, your code from programming assignments, the code on the COS 126 course website, materials from lectures, precepts, and existing Ed posts.

No communication is permitted (e.g., talking, texting, Ed, etc.) during the exam, except with course staff. We will be sitting outside the exam room if you have a question.

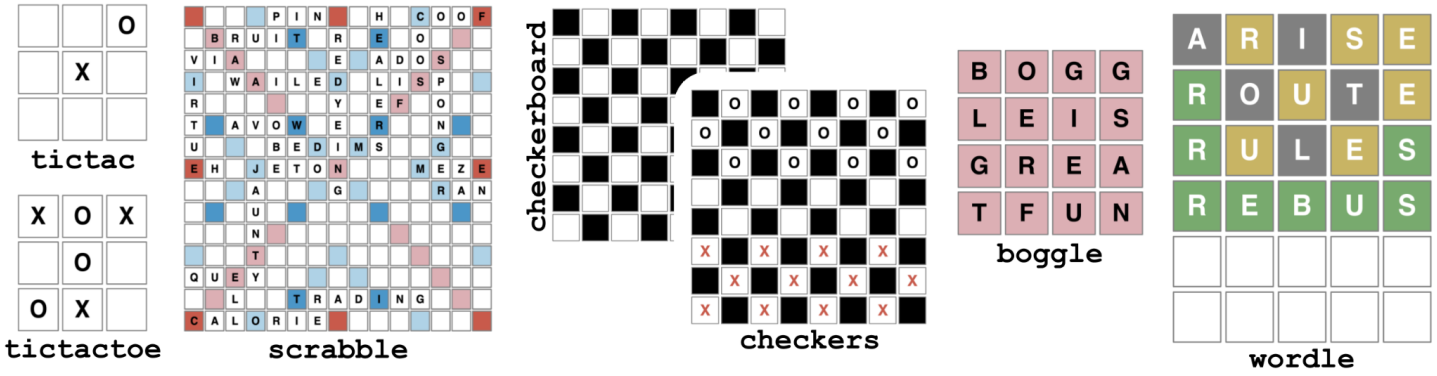
Do not discuss later. Due to multiple exam times, and various conflicts, some of your peers will take the exam at a different time. Do not discuss the exam contents with anyone (not even other students that you know already took the exam!) until after the graded exams are returned.

Honor Code pledge. You must electronically sign the honor code by retyping the pledge below and then /s/ in the file `honor-code.txt`, and upload it to TigerFile as part of your solution. Please do this now. (The /s/ is your "electronic signature.")

"I pledge my honor that I have not violated the Honor Code during this examination. /s/"

Do not examine or start the exam until instructed to do so.

Problem. Develop a Java program that can draw one of several game boards, shown here:



Library. To help you get started, you have access to a new library called **`Board`**, similar to **`StdIn`** and **`StdDraw`**, which provides the following functions:

`public static void` **`initBoard`**`(int rows, int cols)`
Initialize the game board to have the given number of rows and columns, clearing all boxes to empty/white.

`public static void` **`letterBox`**`(int row, int col, char letter, String textColor, String boxColor)`
Draw a letter in the box at the given row and column (both 0 in the upper left corner) using provided color names for the letter and the box. Color names are represented as strings, with the following options:
`"white", "gray", "black", "yellow", "green", "red", "pink", "lightblue", "darkblue"`

Here is an example of a function that draws the picture labeled **`tictac`** in the upper left corner above:

```
// draw a 3x3 tic-tac-toe board with X in the center and O in the upper-right
public static void tictac() {
   Board.initBoard(3, 3);
   Board.letterBox(1, 1, 'X', "black", "white");
   Board.letterBox(0, 2, 'O', "black", "white");
}
```

The **`Board`** library also provides two helper functions that might be useful (but are not strictly necessary) for your project:

`public static char` **`firstLetter`**`(String str)`
This function takes a `String` argument, and returns the first letter of the `String` as a `char`. For example:
`char c = Board.firstLetter("red"); // assigns 'r' to variable c`

`public static char[]` **`wordToLetters`**`(String word)`
This function takes as input a string, and outputs the corresponding sequence of characters as an array, e.g.:
`char[] letters = Board.wordToLetters("red"); // assigns {'r', 'e', 'd'} to variable letters`

Notes: To fill a box in a particular color, but with no letter, simply call `Board.letterBox()` with the space character (`' '`). Also, if your code first calls `Board.letterBox()` to write a letter at a specific location (row, col), and then later calls it again with a different letter (or color) but at the *same* location, the latter will overwrite the former.

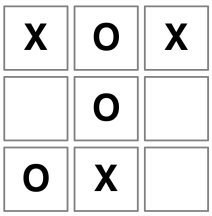
Warnings: There are two possible warnings that you should ignore during this exam: (1) on some Macs, a Java warning about Times font, and (2) IntelliJ's checkstyle warnings about repeated string literals like `"white"`.

Step 0 - `tictac`. In your project folder you are given a class called **`Game.java`**. It already has 3 functions implemented: **`tictac`** (code shown above), **`chooseGameByName`**, and **`main`**. If you compile and then run the program as follows, it will draw the picture shown in the top left corner of this page: `java-introcs Game tictac`

All the functions you need to implement in this exam are already declared in the provided file **`Game.java`**. Do not modify their signatures. You may add your own (`private`) "helper" functions, but it is not needed in order to complete the exam. For this exam you may also assume that all inputs are valid, including function arguments and standard input.

Step 1 - `tictactoe`. Implement this function (**`tictactoe`**) in **`Game.java`** – similar to **`tictac`**, except that it reads the player moves from standard input using three functions from **`StdIn`**: **`readInt`**, **`readString`** and **`isEmpty`**.. The format of the input is in three columns like below – `int`, `int`, `String` representing *row*, *column*, *player*.

```
0 0 X
1 1 O
0 2 X
0 1 O
2 1 X
2 0 O
```



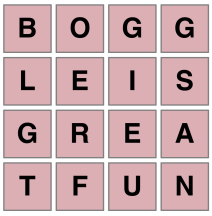
The first line places **X** in the upper-left corner of the board. The second line places **O** in the center. And so forth. *Hint:* in your function you may want to use the function `Board.firstLetter` (see page 2). *Another hint:* do not try to use `StdIn.readChar` – it will not help in this situation. This command draws the picture shown:
`java-introcs Game tictactoe < data-tictactoe.txt`

Step 2 - `scrabble`. Implement this function, which is similar to **`tictactoe`** but also includes a background color:

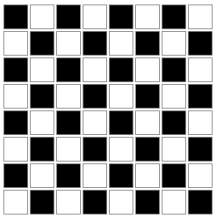
```
0 0 - red
0 3 - lightblue
0 7 - red
[...]
0 4 P white
0 5 I white
0 6 N white
[...]
```

It draws a 15x15 Scrabble board, reading letters and colors from standard input. Each line contains row, col, letter, and background color. (The text color should be `black`.) In the input file, hyphen (`'-'`) is a placeholder for space (`' '`), and is used to draw the colored empty squares, like red in the upper-left corner (first line of example input above). *Hint:* in Java, `char` is a primitive data type, and values of this type can be compared using double equals (==).
This command draws the picture:
`java-introcs Game scrabble < data-scrabble.txt`

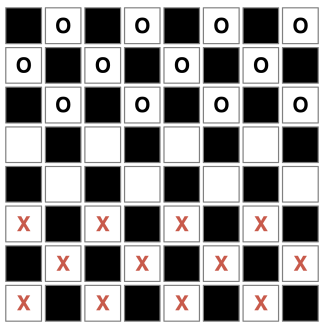
Step 3 - `boggle`. Implement a function that draws a 4x4 grid of `pink` boxes with `black` letters. Instead of reading standard input, this function has one argument – an array of strings. Assume that the array has exactly 4 strings, and each string has exactly 4 letters. For example, it would draw the picture to the right if passed this input array: `["BOGG", "LEIS", "GREA", "TFUN"]`. *Hint:* in your function you may want to use the function `Board.wordToLetters` (see page 2).
This command draws the picture: `java-introcs Game boggle`



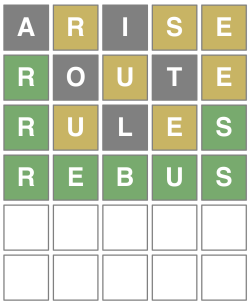
Step 4 - `checkerboard`. Implement this function, which draws an 8x8 board with `black` in the upper-left corner and alternating `black`/`white` from there, as shown in the picture on the right.
This command draws the picture: `java-introcs Game checkerboard`



Step 5 - `checkers`. Implement this function, which first draws the board from Step 4, but also adds player pieces as upper-case letters – `black` **O**'s in the top three rows of `white` boxes, and `red` **X**'s in the bottom three rows, as shown on the right. Hint: it is ok to call your function from Step 4, if you want.
This command draws the picture: `java-introcs Game checkers`



Step 6 - (bonus challenge) `wordle`. *Congratulations if you made it this far! This function is more difficult, to provide an extra challenge for those with sufficient time. This step only accounts for a small portion (<7%) of your exam, so only attempt it if you are confident in your previous steps.*



Read a sequence of 5-letter words from standard input. The first word is the *secret*. Subsequent words are *guesses*. Guesses are displayed on a 6x5 board, one-per row, starting at the top row, as shown in the picture. White letters appear in boxes of one of three colors:

`gray` – This letter is not part of the secret word.
`green` – This letter matches the corresponding secret letter.
`yellow` – This letter is in the secret word at a different location.

In the example, the secret word is REBUS. The first guess (ARISE) had three correct letters, but they were in the wrong locations (yellow). In the next guess (ROUTE) the R is in the correct location (green). In the third row, two letters (R,S) are in the correct location (green), and two others (U,E) are in the secret word, but in a different location (yellow). The fourth row, which is all correct, reveals the secret. You may assume that no letters appear more than once in a word. (Later after the exam, think about how repeated letters might make the problem more difficult, and what you might do about it.)

This command draws the picture: `java-introcs Game wordle < data-wordle.txt`