Project 3

CPSC 2150

Due: Thursday, November 3rd at 11:59 pm

100 pts

Introduction:

We will continue to work on our Extended Tic-Tac-Toe game for this assignment, but we will be adding new requirements. The old requirements are still required unless a requirement in this prompt replaces them.

New Requirements:

- At the start of each new game, the user will provide the number of rows, columns, and markers in a row needed to win the game. These parameters should be passed to the `GameBoard` class in the constructor. The maximum number of rows and columns for all `GameBoard`s is 100, and the maximum number needed to win will be 25 for all `GameBoard`s. The minimum number of rows, columns, and makers in a row needed to win are all 3. `GameBoard.java` must still use a 2D array to represent its game board, but it's ok if most of the array is ignored for smaller boards.
    - The user should not be able to provide a "number of markers needed to win" that is larger than the number of rows or the number of columns.
    - The constructors for the game boards should take parameters in the order of rows, columns, and the number of markers needed to win.
- Note that the formatting of the displayed board has changed to allow double-digit numbers.
- If the users decide to play again, they should be able to change the size and number of markers needed to win settings from the previous game.
- The program must validate the number of rows, columns, and makers needed to win that is provided by the user.
- At the start of each new game, the users will be able to specify the number of players for each game. They must enter at least 2 players and, at most 10 players. Note that `IGameBoard` does not care about the number of players at all; this is just a requirement of `GameScreen`
- After selecting the number of players, each player will be able to pick what character will represent them in the game. You can assume they will always enter just a character. Always convert the character to upper case. Each player must have their own unique character, and the game should not allow a player to pick a character that is already taken.
    - **Note:** if you have a `String`, you can call the `charAt(0)` method to get the first character. If your `String` only has one character, this is an easy and safe way to convert a `String` provided by the user (`scanner.nextLine()` returns a `String`) and convert it to a character.
    - **Note:** The `List` interface has a method called `contains(Object obj)` that will return true if the `obj` already exists in the `List`.
- At the start of each new game, the players can choose whether they want a fast implementation or a memory-efficient implementation. The game must check to make sure they entered either `f`, `F`, `m`, or `M` as a choice at this option.

    - Remember, if you use == to compare `String`, it checks reference equality (i.e., checking to see if the memory locations are the same). Use the `equals()` method to check if they have the same abstract value.
- The number of players and their characters can change if they choose to start a new game.
- The choice of fast or memory-efficient implementation can be changed if they choose to start a new game.
- Our game must now have two different implementations of our `IGameBoard` interface
    - We already have our fast implementation, which uses a 2D array. However, we will now be checking to make sure the implementation is efficient. So, you should not be checking the entire row if you are looking for the horizontal win; you should be checking starting at the last position played.
    - Our memory-efficient implementation is detailed below.
    - Our `GameScreen` class should be able to switch between these two implementations easily. Remember, coding (programming) to the interface allows for the code to be the same except for the call to the constructor. You should not have essentially 2 different versions of the `main` function (one for each implementation) but one `main` method that has an `if` statement to call the appropriate constructor of your `IGameBoard` object.

New Implementation:

You will need to provide a new class called `GameBoardMem` that will extend `AbsGameBoard`, which implements the `IGameBoard` interface. This new implementation will be slower but more memory efficient.

To represent the game board, you will use a `Map`. The `Map` is another generic collection that uses a `key` – `value` pair. The `key` field of our `Map` will be a `Character` that represents the player so that each player will get their own entry in the `Map`. The `value` associated with that player will be a `List` of `BoardPosition`s that the player occupies on the board. If a player has not placed a token, there will be no `key` or `List` in the `Map` to represent that player. Do NOT create a key for the blank space with a `List` of `BoardPosition`s that are blank. The memory efficiency of this implementation comes from the fact that the empty board does not use up any space in memory. When a player adds their first token, you add the key for that player and a `List` with that `BoardPosition` in it. As a player adds more tokens, you add the `BoardPosition` onto the `List` for that player.

There is a video available on Canvas that discusses some `Map`s in more detail and discusses how to declare and work with `Map`s. It describes many useful methods in the `Map` class as well.

While this may seem like a lot of work to create this new implementation, remember that your new implementation only needs to provide code for your primary methods and the constructor. All of your secondary methods can be handled by default implementations in the interface, and `toString` is provided by `AbsGameBoard`. However, we will want to override one of the default implementations.

While this representation of a game board will be more memory efficient, it will be slower. With a 2D array, if we want to know what is in row 2 and column 3, we can directly access that position in the array. However, we can no longer do that with our data representation in a `Map`. If we want to know what's at row 2, column 3, we need to look at each key-value entry in the `Map`. If row 2 and column 3 exist in the `List` (our value in the map), then the player whose character is the key is at that position. If the position does not exist in any list, then the space is blank.

Now consider if instead of wanting to know what's at position (2, 3), we just need to know if 'X' is at position (2, 3). Instead of checking every `List`, we just need to check the `List` with '`X`' as its key. Therefore, our *interface* has two different methods `whatsAtPos` and `isPlayerAtPos`. When we were using a 2D array, they were implemented similarly, but now that we are using a `Map`, the implementations are quite different. Our interface didn't care about how it was implemented; we just knew that asking, "what's at this position?" and "is X at this position?" are different questions and should be asked differently. This means we can override the default `isPlayerAtPos` method in our `GameBoardMem` implementation to more efficiently implement that particular data structure. After doing so, ensure your code calls `isPlayerAtPos` when appropriate and not just rely on `whatsAtPos`.

Example Board:

Consider the following board:

```
    0| 1| 2| 3| 4| 5| 6| 7| 8|
 0|  |  |  |X |X |X |O |  |  |
 1|  |  |  |  |T |  |T |  |  |
 2|  |  |  |O |O |T |  |  |  |
 3|  |  |  |  |  |  |W |  |  |
 4|  |  |  |  |  |  |W |  |  |
```

In our implementation from Project 2, this board would be represented in memory as a 5 by 8 2D array of characters. Blank spaces in the board are represented by that position in the array containing a ' ' character. While we can quickly access any place on the board by jumping to that index in the array, our board takes a lot of memory because it always stores 40 characters. Now consider what this board would look like in our memory-efficient implementation.

In our memory-efficient implementation, we would have a `Map<Character, List<BoardPostion>>` with our `key` being the player that represents a character. We currently only have 4 players (`X`, `O`, `T`, `W`), so those will be the only keys in our `Map`. For our 4 players, we will have `keys` and matching `List`s. So, if we asked our `Map` to return the `value` for `key`: '`W`', it would return a `List` of `BoardPositions (Row, Column)` with `<(3,6), (4,6)>` in it. As player '`W`' places more markers, we can add more `BoardPostions` to the `List`.

Our `key` – `value` pairs for this board would be as follows:

Key	**Value**
X	<(0,3), (0,4), (0,5)>
O	<(2,3), (2,4), (0,6)>
T	<(1,4), (2,5), (1,6)>
W	<(3,6), (4,6)>

Contracts and Comments

All methods (except for the `main`) must have preconditions and postconditions specified in the Javadoc contracts. All methods (except for `main`) must also have the `@params` and `@return` specified

in the Javadoc comments as well. You must include a complete interface specification in `IGameBoard`, and write the contracts not included in the interface specification in the `GameBoard` class. You must provide correspondences and invariants in your `GameBoard` and `GameBoardMem` classes.

Your code must be well commented. The Javadoc comments will only be enough for very simple methods. Remember, your comments will help the grader understand what you are trying to do in your code. If they don't know what you are trying to do, you will lose points.

For this assignment, you should generate the Javadoc documentation using the commands we provided in the lecture 2 slides. An example of what you should generate is under the lecture 2 module on your lecture Canvas page. Place the generated files in a directory called `doc`.

Input and Output

Remember, the TAs will be grading this assignment using scripts as much as possible. You can make this easier on them by using the correct format for input and output. If their scripts don't work because you do not follow the input and output standards, a few things will happen:

1. Grading and feedback will take longer
2. The TA will be mad at you, which you never want when they are grading your assignment
3. The TA will have to spend more time manually using and inspecting your code, which means they are more likely to find minor issues and deduct points.
4. You will lose points for not following the instructions.

At the end of this document, you can see what my input and output look like. **Notice how it handles more than 10 columns and that row and column are asked separately and in that order.**

Project Report.

Along with your "code," you must submit a well-formatted report with the following parts:

Requirements Analysis

Thoroughly analyze the requirements of this program. Express all functional requirements as user stories. Remember to list all non-functional requirements of the program as well. (**Note:** Some of our requirements have changed, so make sure you update those!)

Design

Create a UML class diagram for each class and interface in the program. Also, create UML Activity diagrams for every method in your `GameScreen` class and every method in your `GameBoard` or `GameBoardMem` class. Suppose a method is defined as a `default` method in the interface. In that case, you need to update the diagrams to remove any mention of the 2D array or other private data of the `GameBoard` class and instead refer to primary methods. If a method is provided only as a `default` method in the interface, you only need to provide an activity diagram for the `default` method; you do not need to include the diagram for each implementation. If you provide a `default` method in the interface but override it in the implementation, you must include an activity diagram for each method. Your diagram for `toString` should not reference private data since it will exist in the abstract class.

The activity diagrams for `AbsGameBoard`, `GameBoard`, and `GameScreen` should already be completed from the Projects 1 and 2 submissions, however, once you implement and test your code,

you may discover a mistake. If so, you should correct the error in your code and update the diagrams. As software engineers, we use UML diagrams not just as a design tool but as a way to document our code so other software engineers have reference materials to help them when they need to maintain, fix, test, or build off of our code.

Deployment

You must include a `makefile` with the following targets:

- `default`: compiles your code. Runs with the `make` command.
- `run`: runs your code. Runs with the `make run` command.
- `clean`: removes your compiled (`.class`) files. Runs with the `make clean` command.

Your project report should include instructions on using the `makefile` to compile and run your program.

General Notes and Additional Requirements:

- If it does not compile, it is a zero. Make sure it will compile on SoC Unix machines.
- You will need to place the Java files in the following package structure. `GameScreen` should be placed in `cpsc2150.extendedTicTacToe` and `BoardPosition` `IGameBoard`, `AbsGameBoard`, `GameBoard`, and `GameBoardMem` should be placed under `cpsc2150.extendedTicTacToe.models`
- Remember our "best practices" that we've discussed in class. Use good variable names, avoid magic numbers, etc.
- Start early. You have time to work on this assignment, but you will need time to work on it.
- Starting early means more opportunities to go to TA or instructor office hours for help
- This is an individual assignment; you may not work with a partner.
- There are other ways you could use `Map`s to implement the `IGameBoard` interface, but I am asking for a particular representation. You must use that representation.
- The project report should be one organized document
- You must include a `makefile` with your program. We should be able to run your program by unzipping your directory and using the `make` and `make run` commands.
- Remember, this class is about more than just functioning code. Your design and quality of code are very important. Remember the best practices we are discussing in class. Your code should be easy to change and maintain. You should not be using magic numbers. You should be considering Separation of Concerns, Information Hiding, and Encapsulation when designing your code. You should also be following the idea and rules of design by contract.
- A new game should always start with player 1
- Your code should be well-formatted and consistently formatted. This includes things like good variable names and a consistent indenting style.
- You should not have any "Dead Code" in your code. "Dead Code" is code that is commented out because it is no longer used.
- Your code must be able to run on the SoC Unix machines. Usually, there is no issue with moving code from IntelliJ to Unix, but sometimes there can be issues, especially if you try to import any uncommon Java libraries or use a different version of Java.

- Your UML Diagrams must be made electronically. I recommend the freely available program Diagrams.net (formally Draw.io), but if you have another diagramming tool you prefer, feel free to use that. It may be faster to draw rough drafts by hand, and then when you have the logic worked out, create the final version electronically.
- While you need to validate all input from the user, you can assume that they are entering numbers or characters when appropriate. You do not need to check to see that they entered 6 instead of "six."
- Your code needs to be more than just functioning; it needs to be efficient.
- The `List` interface has a method called `contains`. It will be very helpful. **Note**: it relies on the `equals` method being properly overridden.
- `IGameBoard` does not know how many players are playing the game, and it does not care.

Before Submitting

You need to make sure your code will run on SoC Unix machines and create a `makefile`. Include your name as a comment at the top of each Java file.

Use the command "`zip -r Project3.zip cpsc2150 doc makefile report.pdf`" to zip up all the code, documentation, `makefile`, and project report. Don't forget the `-r` to make sure it includes the subdirectories as well. **NOTE:** Copying and pasting this command may cause issues because Windows encodes the – symbol differently from Unix.

Submission:

Upload your `Project3.zip` to Gradescope. It should automatically check if you have submitted everything correctly and verify that it was able to compile and test your code. While not all test cases are visible, you should make sure that all visible test cases pass. The visible test cases should be a good starting point for checking whether your solution is correct or not.

Your assignment is due at 11:59 pm. Even a minute after that deadline will be considered late. Make sure you are prepared to submit earlier in the day, so you are prepared to handle any last-second issues with submission. **Late submissions will not be accepted.**

This is an INDIVIDUAL assignment. You should not be working on this assignment with any other student.

NOTE: Always check your submissions on Gradescope to ensure you uploaded the correct files and it is in the proper format that it expects. **I wouldn't wait until the last minute to submit as Gradescope could take a few minutes to display the results.** Notify your course instructor immediately of any Gradescope issues.

Disclaimer:

It is possible that these instructions may need to be updated. As students ask questions, I may realize that something is not as straightforward as I thought, and I may add detail to clarify the issue. Changes to the instructions will not change the assignment drastically and will not require reworking code. They would just be made to clarify the expectations and requirements. If any changes are made, they will be announced on Canvas.

Checklist

Use this checklist to ensure you are completing your assignment. **Note:** this list does not cover all possible reasons you could miss points but should cover most of them. I am not intentionally leaving anything out, but I am constantly surprised by some of the submissions we see that are wildly off the mark. For example, I am not listing "Does my program play Tic Tac Toe?" but that does not mean that if you turn in a program that plays Checkers, you can say, "But it wasn't on the checklist!" The only complete list that would guarantee that no points would be lost would be an incredibly detailed and complete set of instructions that told you exactly what code to type, which wouldn't be much of an assignment.

- Can I set the size of my game board?
- Can I set the number needed in a row to win?
- Does my game allow for more than 2 players?
- Do my classes have invariants?
- Did I add an interface?
- Is my interface specification complete?
- Did I add correspondences?
- Does my game allow for players to play again?
- When players choose to play again, can they change the settings of the game?
- Are my methods reasonably efficient? You don't need to take this to the extreme, but obvious inefficiencies should be corrected.
- Does my game take turns with the players?
- Does my game correctly identify wins?
- Does my game correctly identify tie games?
- Does my game run without crashing?
- Does my game validate user input?
- Did I protect my data by keeping it private, except for `public static final` variables?
- Did I encapsulate my data and functionality and include them in the correct classes?
- Did I follow Design-By-Contract?
- Did I provide contracts for my methods?
- Did I comment my code?
- Did I provide Javadoc comments for each method?
- Did I avoid using magic numbers?
- Did I use good variable names?
- Did I follow best practices?
- Did I remove "Dead Code?" Dead Code is old code that is commented out.
- Did I make any additional helper functions I created `private`?
- Did I use the `static` keyword correctly?
- Did I express all functional requirements as user stories?
- Did I create my activity diagrams for all methods?
- Did I create a class diagram for each class and interface?
- Did I use Javadoc to generate my documentation?
- Did I provide a working `makefile`?
- Does my code compile and run on Unix?
- Is my program written in Java?

- Does my output look like the provided examples?

Sample Input and Output:

```
How many players?
12
Must be 10 players or fewer
How many players?
1
Must be at least 2 players
How many players?
4
Enter the character to represent player 1
x
Enter the character to represent player 2
x
X is already taken as a player token!
Enter the character to represent player 2
o
Enter the character to represent player 3
t
Enter the character to represent player 4
F
How many rows?
0
Rows must be between 3 and 100
How many rows?
150
Rows must be between 3 and 100
How many rows?
12
How many columns?
0
Columns must be between 3 and 100
How many columns?
150
Columns must be between 3 and 100
How many columns?
15
How many in a row to win?
5
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
g
Please enter F or M
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
m
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player X Please enter your ROW
4
Player X Please enter your COLUMN
7
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player O Please enter your ROW
6
Player O Please enter your COLUMN
2
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player T Please enter your ROW
```

```
13
Player T Please enter your COLUMN
17
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

That space is unavailable, please pick again
Player T Please enter your ROW
8
Player T Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player F Please enter your ROW
8
Player F Please enter your COLUMN
8
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |  |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player X Please enter your ROW
4
Player X Please enter your COLUMN
6
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |X |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |  |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player O Please enter your ROW
6
Player O Please enter your COLUMN
3
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |X |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |O |  |  |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player T Please enter your ROW
6
Player T Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |X |X |  |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |O |  |T |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player F Please enter your ROW
4
Player F Please enter your COLUMN
8
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |  |O |O |  |T |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player X Please enter your ROW
6
Player X Please enter your COLUMN
1
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |  |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
Player O Please enter your ROW
4
Player O Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player T Please enter your ROW
10
Player T Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |  |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player F Please enter your ROW
5
Player F Please enter your COLUMN
8
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |  |  |  |  |  |  |  |
```

```
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player X Please enter your ROW
6
Player X Please enter your COLUMN
8
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |  |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player O Please enter your ROW
5
Player O Please enter your COLUMN
4
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |O |  |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player T Please enter your ROW
7
Player T Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player F Please enter your ROW
9
Player F Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |F |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player X Please enter your ROW
3
Player X Please enter your COLUMN
5
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |X |  |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |F |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
```

```
Player O Please enter your ROW
3
Player O Please enter your COLUMN
6
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |X |O |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |  |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |F |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player T Please enter your ROW
5
Player T Please enter your COLUMN
7
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |X |O |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |T |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
 7|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
 8|  |  |  |  |  |T |  |  |F |  |  |  |  |  |  |
 9|  |  |  |  |  |F |  |  |  |  |  |  |  |  |  |
10|  |  |  |  |  |T |  |  |  |  |  |  |  |  |  |
11|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Player F Please enter your ROW
7
Player F Please enter your COLUMN
2
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 1|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 2|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
 3|  |  |  |  |  |X |O |  |  |  |  |  |  |  |  |
 4|  |  |  |  |  |O |X |X |F |  |  |  |  |  |  |
 5|  |  |  |  |O |  |  |T |F |  |  |  |  |  |  |
 6|  |X |O |O |  |T |  |  |X |  |  |  |  |  |  |
```

```
 7|   |   |F |   |   |T |   |   |   |   |   |   |   |   |   |
 8|   |   |   |   |   |T |   |   |F |   |   |   |   |   |   |
 9|   |   |   |   |   |F |   |   |   |   |   |   |   |   |   |
10|   |   |   |   |   |T |   |   |   |   |   |   |   |   |   |
11|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Player X Please enter your ROW
2
Player X Please enter your COLUMN
4
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 1|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 2|   |   |   |   |X |   |   |   |   |   |   |   |   |   |
 3|   |   |   |   |   |X |O |   |   |   |   |   |   |   |
 4|   |   |   |   |   |O |X |X |F |   |   |   |   |   |
 5|   |   |   |   |O |   |   |T |F |   |   |   |   |   |
 6|   |X |O |O |   |T |   |   |X |   |   |   |   |   |
 7|   |   |F |   |   |T |   |   |   |   |   |   |   |   |
 8|   |   |   |   |   |T |   |   |F |   |   |   |   |   |
 9|   |   |   |   |   |F |   |   |   |   |   |   |   |   |
10|   |   |   |   |   |T |   |   |   |   |   |   |   |   |
11|   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Player O Please enter your ROW
2
Player O Please enter your COLUMN
7
Player O wins!
    0| 1| 2| 3| 4| 5| 6| 7| 8| 9|10|11|12|13|14|
 0|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 1|   |   |   |   |   |   |   |   |   |   |   |   |   |   |
 2|   |   |   |   |X |   |   |O |   |   |   |   |   |   |
 3|   |   |   |   |   |X |O |   |   |   |   |   |   |   |
 4|   |   |   |   |   |O |X |X |F |   |   |   |   |   |
 5|   |   |   |   |O |   |   |T |F |   |   |   |   |   |
 6|   |X |O |O |   |T |   |   |X |   |   |   |   |   |
 7|   |   |F |   |   |T |   |   |   |   |   |   |   |   |
 8|   |   |   |   |   |T |   |   |F |   |   |   |   |   |
 9|   |   |   |   |   |F |   |   |   |   |   |   |   |   |
10|   |   |   |   |   |T |   |   |   |   |   |   |   |   |
11|   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Would you like to play again? Y/N
y
How many players?
3
Enter the character to represent player 1
```

```
w
Enter the character to represent player 2
y
Enter the character to represent player 3
r
How many rows?
5
How many columns?
5
How many in a row to win?
3
Would you like a Fast Game (F/f) or a Memory Efficient Game (M/m?
f
    0| 1| 2| 3| 4|
 0|  |  |  |  |  |
 1|  |  |  |  |  |
 2|  |  |  |  |  |
 3|  |  |  |  |  |
 4|  |  |  |  |  |

Player W Please enter your ROW
…
```