CS 142 Major Assignment 4
Crickets and Grasshoppers

In this assignment, you'll write an interactive, 2-player game called "Crickets and Grasshoppers" that is played through the console. This page contains the rules of the game, and later pages will detail the assignment parameters.

Game Rules:

Crickets and Grasshoppers is a simple two player game played on a strip of *n* spaces. Each space can be empty or have a piece. The first player plays as the cricket pieces and the other plays as grasshoppers and the turns alternate. We'll represent crickets as C and grasshoppers as G. Crickets start on the left side moving right and grasshoppers start on the right side moving left.

Each turn, a player **must** move one of their pieces. A piece can either move forward one space if it is empty, or jump over one of the opponent's pieces immediately in front of it, landing on an empty space right after the opponent's piece.

The number of pieces never changes (pieces are never captured or added).

A player wins if their opponent has no possible moves on their turn.

Examples:

Here are three example playthroughs, with user input bold and underlined. So that the tester can accurately test your code, **your program should match this behavior exactly, including the wording**.

```
Please enter the number of pieces for each
player (1-10): 2
Please enter the number of spaces in the
middle (1-9): 1
CC.GG
Crickets, please enter your move (1-5): 2
C.CGG
Grasshoppers, please enter your move (1-5): 4
CGC.G
Crickets, please enter your move (1-5): 3
CG.CG
Grasshoppers, please enter your move (1-5): 5
CGGC.
Crickets, please enter your move (1-5): 4
Crickets win!
```

```
Please enter the number of pieces for each
player (1-10): 2
Please enter the number of spaces in the
middle (1-9): 2
CC..GG
Crickets, please enter your move (1-6): 2
C.C.GG
Grasshoppers, please enter your move (1-6): 5
C.CG.G
Crickets, please enter your move (1-6): 1
.CCG.G
Grasshoppers, please enter your move (1-6): 5
That space does not contain a piece you can
move! Please try again.
Grasshoppers, please enter your move (1-6): 6
Grasshoppers win!
```

```
Please enter the number of pieces for each
player (1-10): 2
Please enter the number of spaces in the
middle (1-9): 2
CC..GG
Crickets, please enter your move (1-6): 2
C.C.GG
Grasshoppers, please enter your move (1-6): 5
C.CG.G
Crickets, please enter your move (1-6): 3
C..GCG
Grasshoppers, please enter your move (1-6): 4
C.G.CG
Crickets, please enter your move (1-6): 1
.CG.CG
Grasshoppers, please enter your move (1-6): 0
That was not a valid number! Please try again.
Grasshoppers, please enter your move (1-6): 3
GC..CG
Crickets, please enter your move (1-6): 2
G.C.CG
Grasshoppers, please enter your move (1-6): 6
G.CGC.
Crickets, please enter your move (1-6): 5
G.CG.C
Grasshoppers, please enter your move (1-6): 4
GGC..C
Crickets, please enter your move (1-6): 3
Crickets win!
```

Requirements:

Please create a class called **`CricketsAndGrasshoppers`** that implements the following methods.
You may write additional helper methods if desired.

`public static int promptNumberReadLine(Scanner s, String prompt, int max)`

Print the `prompt` using `System.out.print()` (just `print`, not `println`). If the next piece of information in the Scanner represents an integer which is at least 1 and at most `max`, read in the rest of the line with `s.nextLine()` and then return the number.
Otherwise (if the next piece of information doesn't meet the above requirements), read in the rest of the line with `s.nextLine()`, and print the line:
`That was not a valid number! Please try again.`
then repeat the process described in this box. **Follow these directions precisely for full points.**

`public static int[] createBoard(int piecesPerPlayer, int spacesInMiddle)`

Create and return an array representing a new game with the number of pieces for each player indicated. The pieces should be on the ends of the board with the specified empty spaces.
In your array, use `1` to represent Crickets (player 1), use `2` to represent Grasshoppers (player 2), and use `0` to represent an empty space.

`public static String boardToString(int[] board)`

Create and **return** a String that represents the game board, all on one line. Crickets are specified with C, grasshoppers with G, and empty spaces with . (period)
Don't print it with this method! You'll print the board by calling this method from the `main` later.
Hint: Use string concatenation in a `for` loop.

`public static boolean canMove(int[] board, int player)`
Return `true` if the given `player` has any move they can make. Cricket is player 1 and grasshopper is 2.
This method will help you determine when the game is over.

`public static boolean move(int[] board, int player, int position)`

The `player` moves their piece in the given position **(numbered 1 through *n*)**. If the specified move is allowed, modify the `board` and return `true`. Otherwise, don't modify board and return `false`.
Note: The user will enter a number form `1` to `n` to specify a `position`, but in your `board` array, recall that the entries are indexed `0` to `n-1`.

`public static void main(String[] args)`

The main method will do several things in this program:
Start by creating a new Scanner object, then use your `promptNumberReadLine` method to ask the user for the number of pieces per player and empty spaces in the middle. Use this information to create the initial board.
Then use a while loop to repeat the primary game loop where you:
- print the board to the console as a String,
- ask the current player to enter their move, and, if it is valid,
- modify the board with your `move` method. If it is not valid, print a message and ask again.

Repeat this until a player has won.
Once a player has won, exit the loop and print out a message as shown in the examples above.

Output Guidelines:

Your program should begin by asking how many pieces each player has with a prompt as follows, with 10 as the maximally allowed number of pieces (user input shown **bold and underlined**):

```
Please enter the number of pieces for each player (1-10): two
```

To do this, use the `promptNumberReadLine` method. If the user does not type a number in the correct range, that method will prompt them in this way:

```
That was not a valid number! Please try again.
Please enter the number of pieces for each player (1-10): 2
```

Next, again with `promptNumberReadLine`, ask how many spaces should be in the middle with 9 as the maximum:

```
Please enter the number of spaces in the middle (1-9): 1
```

Before each move, display the current state of the game by printing out the result of `boardToString`, then ask the player which position to move (1 through *n*) using the `promptNumberReadLine` method. Please also note the error messages and re-prompting below. **Your messages must match these exactly.**

```
CC.GG
Crickets, please enter a position to move (1-5): 1
That space does not contain a piece you can move! Please try again.
Crickets, please enter a position to move (1-5): 4
That space does not contain a piece you can move! Please try again.
Crickets, please enter a position to move (1-5): 7
That was not a valid number! Please try again.
Crickets, please enter a position to move (1-5): 2
C.CGG
Grasshoppers, please enter your move (1-5): 4
CGC.G
Crickets, please enter your move (1-5):
```

Note that during the first move, only positions 1 and 2 contain the cricket pieces (and only the piece in position 2 can be moved), but as moves are made, other positions could contain crickets, so we always include all position numbers in the prompt.

Also notice that there are two (and only two) different error messages:

- The “valid number” error message is given if the number was not in the correct range. This should be provided by the `promptNumberReadLine` method.
- The “does not contain” error message is given if a position that can’t move is entered, whether that position contains a trapped player piece, an opponent piece, or a space. This should be done in the `main` method.

A final warning: Only the `promptNumberReadLine` and `main` methods should print!