CS300: PROGRAMMING II
Department of Computer Sciences

PEG SOLITAIRE GAME

This assignment has been designed as a self-assessment for students considering enrollment in CS300. Our expectation is that well prepared students should be comfortable completing an assignment like this within two weeks (or about 15 hours, since CS300 is a 3 credit class).

Although students will use Java to complete their CS300 assignments, this self-assessment can be completed in whatever language you feel most comfortable working. Students with sufficient experience in languages like: C, C++, Javascript, Python, Matlab, and others can expect to find many familiar data types and control structures in Java. Experience with designing, implementing, testing, and debugging code written in these kinds of languages should transfer nicely toward developing code in Java. You'll note in the required organization below that your solution will be in a more procedure-oriented rather than object-oriented form. This is because the later organizational paradigm will be taught in CS300.

The rest of this document is organized into the following sections:

- The Rules of Peg Solitaire – The rules of the game you are implementing for this assignment.
- Organizational Requirements – How your solution to this assignment must be organized.
- Sample Runs – Demonstrate the output of a working implementation.
- Code Templates – Files containing method stubs, ready for you to implement.

THE RULES OF PEG SOLITAIRE

Peg solitaire is a board game (or puzzle) in which a single player takes turns removing pegs from a board until either: they have removed all but the final peg to win the game, or they have lost because there are no legal moves left for them to make despite more than one peg remaining on the board. The number of pegs and board-shapes vary, but these pegs are always set into holes arranged in a grid on the board. A legal move involves jumping one peg over a neighboring peg to rest in a hole on the other side, and then removing the peg that was jumped over. Diagonal jumps are not allowed. In the following examples, an at sign (@) is used to represent the position of pegs, and a dash (-) is used to represent the position of an empty holes.

```
@@-
@@@
@@@
```

(turn1)

```
--@
@@@
@@@
```

(turn2)

```
@-@
-@@
-@@
```

(turn3)

```
@-@
-@@
@--
```

(turn4)

```
@--
-@-
@-@
```

In the example above, the first turn involves moving the peg from the upper-left corner of the board, over the peg in the top-middle position, and landing in the hole in the upper-right corner. The peg in the top-middle position is then removed, since it was jumped over. Can you follow the subsequent moves starting with the lower-left peg on turn 2, lower-right peg on turn 3, and upper-right peg on turn 4? After this sequence of moves, there are no more legal moves to be made. And since there is more than one peg left on the board, this example demonstrates a lost game.

More information about this game is available on Peg Solitaire Game on Wikipedia.

ORGANIZATIONAL REQUIREMENTS

Although you are free to implement this assignment in a language of your choice, well prepared CS300 students are comfortable organizing their code into methods (also called functions, procedures, subroutines, etc). You are allowed to make use of global (or static) constants, but all mutable variables should be defined locally within some method. All communication between methods must be passed through their arguments and return values. To demonstrate your ability to implement this game with a procedure oriented organization, your code for this assignment must be organized into the following methods.

- void main(String[] args) // drives the entire game application
- int readValidInt(Scanner in, String prompt, int min, int max) // returns a valid integer from the user
- char[][] createBoard(int boardType) // returns an array, initialized according to a specific board type
- void displayBoard(char[][] board) // prints out the contents of a board to for the player to see
- int[] readValidMove(Scanner in, char[][] board) // reads a single peg jump move in from the user
- boolean isValidMove(char[][] board, int row, int column, int direction) // checks move validity
- char[][] performMove(char[][] board, int row, int column, int direction) // applies move to a board
- int countPegsRemaining(char[][] board) // returns the number of pegs left on a board
- int countMovesAvailable(char[][] board) // returns the number of possible moves available on a board

Note that these method headers correspond to their Java implementations, and may differ from other language in some ways. The Scanner parameters in the methods above are used to read input from the user through standard-in. Although a parameter like this may not be necessary in some languages, seeing this parameter still provides a helpful hint about which methods may require extra information from the user. The other types are likely to look more familiar: int is an integer numeric type, char[][] is the type for a random access two dimensional array of characters, and void means that no value is returned from a method. Another thing worth noting is that the size of an array is accessible through an array reference in Java. If this is not the case in your language, additional parameters may be necessary to communicate this information.

SAMPLE RUNS

The user's input in each of the following sample runs is displayed as bold orange text.

Winning Simple T Run

```
WELCOME TO CS300 PEG SOLITAIRE!
===============================

Board Style Menu
  1) Cross
  2) Circle
  3) Triangle
  4) Simple T
Choose a board style: 4

  12345
1 -----
2 -@@@-
3 --@--
4 --@--
5 -----
Choose the COLUMN of a peg you'd like to move: 3
Choose the ROW of a peg you'd like to move: 2
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 4

  12345
1 -----
2 -@--@
3 --@--
4 --@--
5 -----
Choose the COLUMN of a peg you'd like to move: 3
Choose the ROW of a peg you'd like to move: 4
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 1

  12345
1 -----
2 -@@-@
3 -----
4 -----
5 -----
Choose the COLUMN of a peg you'd like to move: 2
Choose the ROW of a peg you'd like to move: 2
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 4

  12345
1 -----
2 ---@@
3 -----
4 -----
5 -----
Choose the COLUMN of a peg you'd like to move: 5
Choose the ROW of a peg you'd like to move: 2
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 3

  12345
1 -----
2 --@--
```

```
3 -----
4 -----
5 -----
Congrats, you won!

==========================================
THANK YOU FOR PLAYING CS300 PEG SOLITAIRE!
```

Losing Simple T Run

```
WELCOME TO CS300 PEG SOLITAIRE!
===============================

Board Style Menu
  1) Cross
  2) Circle
  3) Triangle
  4) Simple T
Choose a board style: 4

  12345
1 -----
2 -@@@-
3 --@--
4 --@--
5 -----
Choose the COLUMN of a peg you'd like to move: 3
Choose the ROW of a peg you'd like to move: 3
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 1

  12345
1 --@--
2 -@-@-
3 -----
4 --@--
5 -----
It looks like there are no more legal moves.  Please try again.

==========================================
THANK YOU FOR PLAYING CS300 PEG SOLITAIRE!
```

Bad Input Triangle (Partial) Run

```
WELCOME TO CS300 PEG SOLITAIRE!
===============================

Board Style Menu
  1) Cross
  2) Circle
  3) Triangle
  4) Simple T
Choose a board style: three
Please enter your choice as an integer between 1 and 4: 3!!!
Please enter your choice as an integer between 1 and 4: 3

  123456789
```

```
1 ###-@-###
2 ##-@@@-##
3 #-@@-@@-#
4 -@@@@@@@-
Choose the COLUMN of a peg you'd like to move: -2 4 5
Please enter your choice as an integer between 1 and 9: five
Please enter your choice as an integer between 1 and 9: 4
Choose the ROW of a peg you'd like to move: 10
Please enter your choice as an integer between 1 and 4: 2
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 3
Moving a peg from row 2 and column 4 LEFT is not currently a legal move.

Choose the COLUMN of a peg you'd like to move: 5
Choose the ROW of a peg you'd like to move: -1
Please enter your choice as an integer between 1 and 4: 1
Choose a DIRECTION to move that peg 1) UP, 2) DOWN, 3) LEFT, or 4) RIGHT: 0
Please enter your choice as an integer between 1 and 4: down
Please enter your choice as an integer between 1 and 4: 2

  123456789
1 ###---###
2 ##-@-@-##
3 #-@@@@@-#
4 -@@@@@@@-
```

(note that the above sample run is not complete)

CODE TEMPLATES

Files containing stubs for the required methods are available in the following languages:

- Java: PegSolitaireGame.java
- Python: peg_solitaire_game.py