CSC301 – Data Structures 2 – Final Project Deliverable #1

Project Description

The final project consists of two separate deliverables. The first is a puzzle application (a game) while the second is a solver for the puzzle. The puzzle application is worth 30 points and the solver is worth the other 70 points. Each deliverable has its own deadline. The puzzle itself is described in its own section below.

Individual Work

Remember that as explained in the syllabus, collaboration on the final project is restricted. You may discuss the problem with other students and share ideas but those ideas cannot include any code. You are not allowed to share code with other students or to suggest code to other students.

IMPORTANT: Differences from regular homework

- **No Resubmit Folder**
  There will be no late submission / resubmission folder on any of the project deliverables.

- **Late Policy**
  Similar to regular homework assignments, the submission folders for project deliverables will remain open for a limited time after the original due date. Like homework, the penalty for late submission is 2% per hour the submission is late (0.6 points per hour late on the first deliverable and 1.4 points per hour late on the second deliverable). The submission folder will remain open for an additional 48 hours after the due date to accept late submissions. However, this means that a submission that is a full day late will lose 50% of the points and no points are possible once 2 days have passed. Please plan your time accordingly so that you can submit on time.

- **Allowed changes to classes**
  You may add additional methods to the classes you were given and you may have them implement Comparable if you wish. You may also add other classes (that you write) in their own separate files. Just make sure to include all additional files when you make you submission so that I can run your code.

- **Allowed imports**
  You may import and use Java's Random class as well as Java's versions of symbol tables (Java's various Map and Set classes.)

The Game/Puzzle

The game/puzzle consists of a 5 by 5 grid of cells. Each cell could be empty or could contain an 'X'. The object of the game/puzzle is to arrange the X's around the outer edges of the grid (the 16 cells that make up the bottom and top row and the leftmost and rightmost columns). You can shift the cells

around by shifting an entire row left or right or by shifting an entire column up or down. Cells that shift off the end wrap around to the other end. Below are the full details about the 20 moves available to a player:

- a-e: Shift an entire row to the left ('a' shifts the topmost row and e the bottommost row).
- A-E: Shift an entire row to the right ('A' shifts the topmost row and E the bottommost row).
- v-z: Shift an entire column down ('v' shifts the leftmost column and z the rightmost column).
- V-Z: Shift an entire column up ('v' shifts the leftmost column and z the rightmost column).

The player continues to make moves until either they win by arranging the X's around the edges of the grid or they lose by repeating a board position. This is probably easier to understand via a demonstration. A demonstration of how the game is played will be given in class, so please make sure to take a look at the lecture recording if you did not attend the class meeting where the demonstration was given.

Deliverable #1 (30 points)

The first deliverable has three related components:

- **(10 points)** You must implement the Board class given to you in the zip file for the project. It must behave as described in the Javadoc comments included in the skeleton file you received. Make sure to read through those comments so that you know what is required. Your Board class will be tested using the P1Test.java file that was also included in the zip file for the project. Like the homework, each test has a number in its name indicating how much each test is worth. The test points add up to a total of 10 points.

- **(10 points)** You must implement the Game class given to you in the zip file for the project. The only required method is main. When the Game class is executed, the user should be able to play one game of the puzzle.
    1. The game begins by prompting the user for a difficulty level between 1 and 10. The application should then generate a random board by starting from a solved puzzle and then making that many random moves. (Note that this is one of the constructors you are required to implement in the board class). It is possible that after making random moves, the board is actually in the solved configuration. You must continue regenerating the board by making the requested random moves until the board you end up with is not solved.
    2. After displaying the board on the screen, the game proceeds by repeatedly prompting the player for a move and displaying the updated board on the screen. Each of these prompts and responses is called a round.
    3. The rounds continue until either the player wins the game by reaching the solved board configuration (all the Xs are on the outer edges of the board) and the game displaying a message indicating the player has won or the player loses by reaching a repeated board configuration and the game displaying a message indicating that the player has lost.

  Again, a demonstration of the expected behavior will be provided in class, so make sure to watch it to see what is required. Your grade will be based on how closely your game resembles

the one in the demonstration, so once you get your game working, spend some extra time trying to get your output to look as much as possible like the output from the demonstration. You must make use of the Board class in your application.

- **(10 points)** You must answer the questions in the file "DesignDocumentation1.docx". make sure to submit your edited DesignDocumentation1.docx file along with your Java source files in the submission folder on D2L. Your score on the questions depends on how well you explain what your code does as well as the actual decisions you made in your solution. Note that a longer explanation is NOT better. You should be able to answer the questions with just a couple of sentences. Also, if a question asks about the runtime of a given feature or piece of code, don't just say "linear". You must say "linear with respect to _____" and fill in the blank. When we've discussed runtimes of data structures in class, it was always with respect to the size of the data structure (the number of elements in the data structure). This project is about a puzzle/game. It's not at all obvious what the "size" here is.

- You must include a screenshot of what happens when you run P1Test.java. Failure to include the screenshot will result in a 2-point deduction.

Submission

Your submission must include all of the following:

- The DesignDocumentation1.docx file where you answered questions.
- A screenshot of what happens when you run P1Test.java
- All of the files necessary to run your solution. This includes Game.java and Board.java. If you created other classes/files for your solution, make sure to submit them as well. (Do NOT submit P1Test.java as you should not have made any changes to it.)

Tips

- For user keyboard input, you should look at the book's StdIn class. It is already part of the workspace you created for the class. You just need to import it. It was used in the TemperatureTracker example we did toward the beginning of the year. I would stick to using only the readLine() method from StdIn. The readInt method may not work as you expect, especially if you mix number and text input.

- A toString method for the board class is not required, but could be very useful. Not only you use it to display the board to the player while playing the game, but you can use it in the debugger to "see" what a Board object looks like when you click on a Board variable.