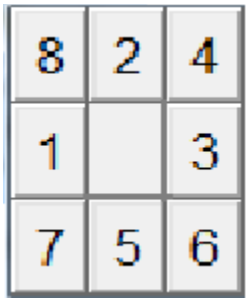
CSC 242 Section 504 Winter 2017
Homework Assignment 6

Due date as specified on D2L

This assignment is worth 3% of your overall grade, and therefore will be graded on a scale of 0-3. Please upload a file containing your competed code onto D2L by the due date. While you may discuss the assignment with others, **you must write your code by yourself,** without assistance from other students or anyone else. Please see the course syllabus for details, and for my late homework submission policy.

The 8-puzzle game

Your assignment is to write a GUI-based program that allows the user to play the 8-puzzle game. In this game, there are 8 tiles that can slide horizontally or vertically on a 3 x 3 board. One of the 9 spaces on the board is blank. The rules are quite simple: the player can slide a tile from its current position to where the blank is provided the tile is adjacent to the blank. For example:



Currently we could move the 1, 2, 3, or 5 tile to the center. If we chose to move the 1, the result would be

From here we could move the 1, 7, or 8.

The player has won when the following board state has been reached:

What I am providing

You may feel free to use my **Board** class, which represents the state of an 8-puzzle board. It is **not** a GUI. The state of the game is represented as follows:

- An instance variable called `blank` represents the current position of the blank spot on the board. It is a list `[ r, c ]` where r is the row of the blank space (0, 1, or 2) and c is the column.
- An instance variable called `positions` represents the current positions of the numbered tiles. This variable is a list. `self.positions[0]` Trepresents the position of the empty space on the board. `self.positions[i]` represents the location of tile number *i*. The items in this list are themselves lists of length 2 `[ r, c ]` which represent the position of each tile. For example: in the initial diagram positions[8] is [0, 0].
- A **goal** method, which returns **True** if the board is in the winning state, or **False** otherwise.
- A **move** method. It is passed the number of a tile and attempts to move that file into the blank position of the board. The tile can only be moved if it is adjacent to the blank position. If it is not, **move** does not do anything. The method returns **True** if a tile has been moved or **False** otherwise.

I have also written part of the constructor for the **EightPuzzle** class. This class is intended to be the GUI portion of the program. In the part that I have implemented, the user is shown a window which asks the user to choose between three skill levels: "easy", "medium", and "hard".

Your task

You program should do each of the following. Each is worth one point.

1. Set up and display a board which is "easy", "medium", or "hard" to solve. In order to do this, your program should call the Board constructor (which creates a Board in the winning state) and make a certain number of moves (thereby taking it out of the winning state). I suggest a small number of moves, such as 5, for "easy", 10 moves for "medium" and 25 for "hard". Then you can present this Board to the user as the starting position.

One would think we could just randomly place the eight tiles in random locations on the board, and present the user with this initial board. However, only about ½ of boards produced in this fashion are solvable without the use of a hammer and/or screwdriver, which are not allowed.. Since we would like for the user to be able to solve the board we present them, we must initialize the board by starting with the goal state and make moves away from the goal state until we have a puzzle with the degree of difficulty that the user asked for.

2. Render the board. In other words, given a **Board** object, display a GUI which reflects the positions of the tiles (and the blank).
3. When the user clicks on one of the tiles, your program should move it if possible. For example, if the user clicks on the "1" in the first diagram, your GUI should move it to the center as shown in diagram 2. However, if the user clicks on the "4" in the first diagram, nothing should happen. You can use the **Board**'s move method to do the internal bookkeeping of which tile is where. The **Board** move method will also return True or False indicating whether a tile has been moved, which can be useful information as you re-draw the board after the move.

For 1 point of extra credit:

4. Your program should recognize when the player has won, display an appropriate response, and ask the user if they want to play again.

The rest of the **EightPuzzle** class is for you to implement.