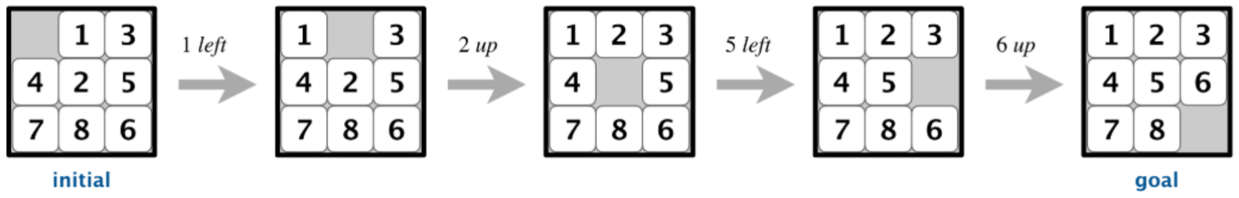
Computer Science 2920

Assignment 1 - Due Friday February 8 @11:55pm

The 8-puzzle is a sliding puzzle that is played on a 3-by-3 grid with 8 square tiles labeled 1 through 8, plus a blank square. The goal is to rearrange the tiles so that they are in row-major order, using as few moves as possible. You are permitted to slide tiles either horizontally or vertically into the blank square. The following diagram shows a sequence of moves from an initial board (left) to the goal board (right).



1. To begin, create a data type that models an n-by-n board with sliding tiles. Implement data type Board with the following API:

```
public class Board {
    public Board(int[][] tiles)           // create a board from an n-by-n array of tiles,
                                          // where tiles[row][col] = tile at (row, col)
    public String toString()              // string representation of this board
    public int tileAt(int row, int col)   // tile at (row, col) or 0 if blank
    public int size()                     // board size n
    public boolean isGoal()               // is this board the goal board?
    public boolean equals(Object y)       // does this board equal y?
    public Iterable<Board> neighbors()    // all neighboring boards
}
```

Constructor. You may assume that the constructor receives an n-by-n array containing a permutation of the n^2 integers between 0 and $n^2 - 1$, where 0 represents the blank square. You may also assume that $2 \leq n \leq 5$.

String representation. The toString() method returns a string composed of n lines. Each line contains the n-by-n grid of tiles in row-major order, using 0 to designate the blank square.

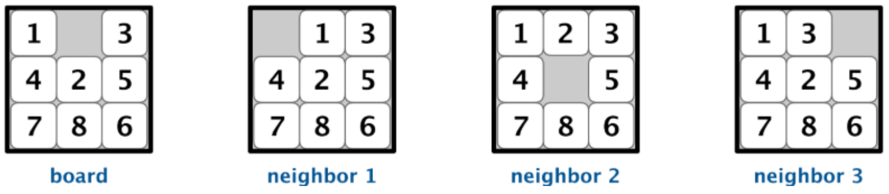
board

```
1  0  3
4  2  5
7  8  6
```
string representation

Tile extraction. Throw a java.lang.IllegalArgumentException in tileAt() unless both row and col are between 0 and n − 1.

Comparing two boards for equality. Two boards are equal if they have the same size and their corresponding tiles are in the same positions. The equals() method is inherited from java.lang.Object, so it must obey all of Java's requirements.

Neighboring boards. The neighbors() method returns an iterable containing the neighbors of the board. Depending on the location of the blank square, a board can have 2, 3, or 4 neighbors.

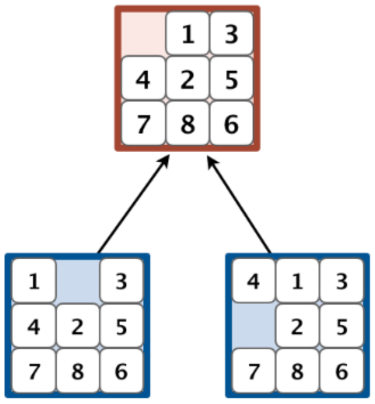


Performance requirements. In the worst case, your implementation must support size() and tileAt() in constant time; the constructor, isGoal(), equals(), toString(), and neighbors() in time proportional to n^2 (or better).

2. Create a game tree data type. In this part you will implement a GameTree class with the following API:

```
public class GameTree {
  public GameTree(Board initial, int depth)// creates a game tree with the given depth
  public String DFS()                      // traverses the tree using DFS
  public String BFS()                      // traverses the tree using BFS
}
```

Constructor. The constructor receives the initial board and creates a game tree where the branches represent the possible movements to perform at a given board. The tree should be created up to the specified depth. Depth = 1 means only the root node, depth = 2 means root node plus one level neighbor boards representing the results of the applicable moves….and so on.



For this game tree DFS and BFS return the same String:

```
013
425
786

103
425
786

413
025
786
```

Depth First Search (DFS). This method will return the sequence of string representations of the boards generated by this traversal algorithm (an empty line should be added between boards).

Breadth First Search (DFS). This method will return the sequence of string representations of the boards generated by this traversal algorithm (an empty line should be added between boards).

3. Create a simple test. Write a program that creates a game tree of depth 3 starting with the board [0 1 3; 4 2 5; 7 8 6] and runs and prints the output of both search methods.

Submission Guidelines:

Submit your assignment as a single zip file through the moodle link provided. Use the following naming format for your zip file:

lastname-as1.zip

The zip file should contain the code of your solution.