CSC 225 - FALL 2015
ALGORITHMS AND DATA STRUCTURES I
PROGRAMMING ASSIGNMENT 5
UNIVERSITY OF VICTORIA

Due: Friday, December 4th, 2015 at 4:00pm.

1 Programming Assignment

The *9-puzzle* consists of a square grid containing eight tiles, marked with the numbers 1 through 8. One of the spaces in the grid is empty. The initial state of the puzzle is the configuration below:

1	2	3
4	5	6
7	8	

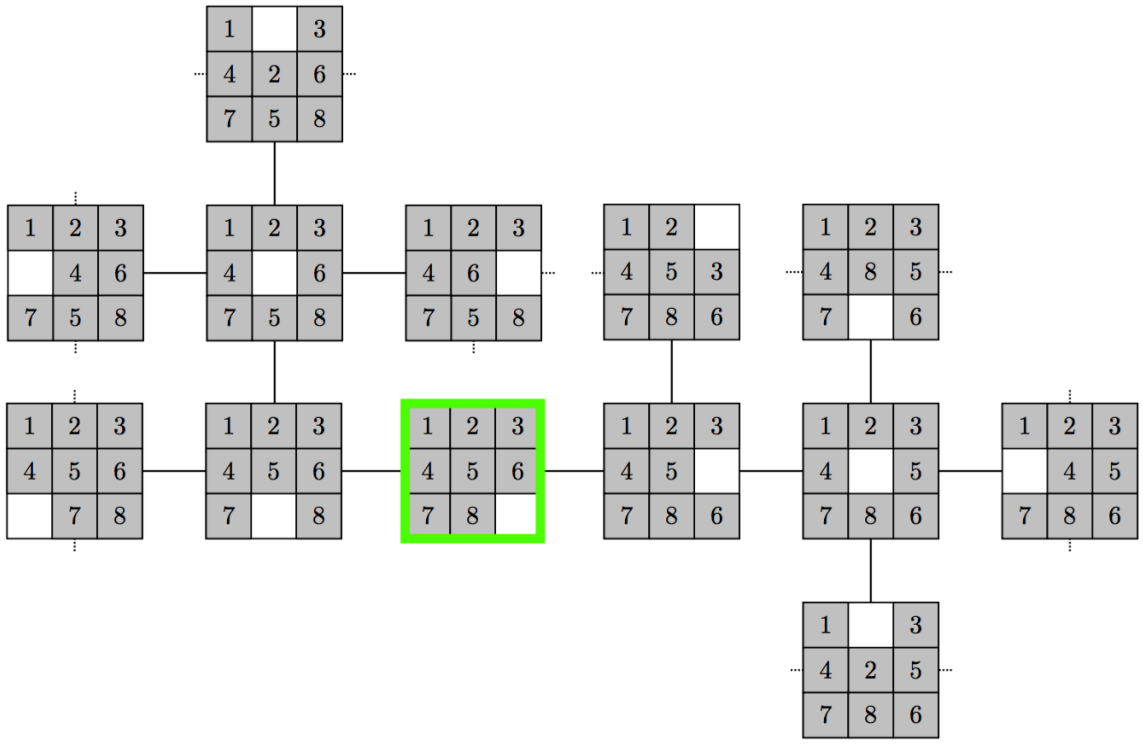
This is considered to be the 'solved' state of the puzzle and is normally called the 'goal state'. The tiles on the board can be moved horizontally or vertically into the empty space to scramble the ordering of the board, as in the configuration below:

4	1	3
	2	6
7	5	8

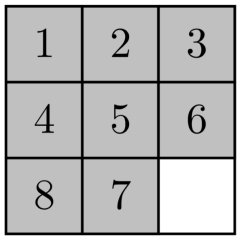
The programming assignment is to implement a solver for the 9-puzzle. Your submission will read a sequence of boards and, for each board, output a sequence of moves which solves the board. The information about the puzzle given here is meant to be a basic summary; for more detailed information, see the additional notes in the resources tab of conneX[1].

[1]The Wikipedia page `http://en.wikipedia.org/wiki/15_puzzle` also contains some helpful information about puzzles of this type.

The different configurations of the 9-puzzle and their relationships can be modelled with a graph. Each vertex corresponds to a possible configuration of the board. Edges represent the transformations possible by making one (legal) move. That is, if two different board configurations are connected by an edge, there is a way to obtain one configuration from the other by making a single move. To solve a given board, tiles are moved until the goal state is reached. The diagram below shows a small snapshot of the graph, with the goal state framed in green.



The set of moves needed to solve a given board is represented by a path in the graph from the board to the goal state. Therefore, a board can be solved by performing one of the two fundamental graph traversals -DFS or BFS- on the graph and searching for a path to the goal state. Some possible board configurations cannot be solved, such as the following:



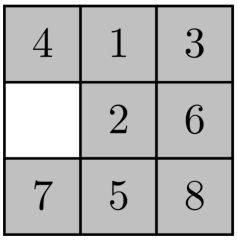
A Java template has been provided containing an empty method `SolveNinePuzzle`, which takes a 3×3 integer array B as its only argument. The expected behaviour of the method is as follows:

Input:	A 3×3 array B representing a 9-puzzle board.
Output:	`true` if B is solvable and `false` otherwise.
Side Effects:	If the board is solvable, a sequence of boards will be printed representing each step of the solution (starting with the initial board and ending with the solved board.

Your task is to write the body of the `SolveNinePuzzle` method. You must use the provided Java template as the basis of your submission, and put your implementation inside the `SolveNinePuzzle` method in the template. You may not change the name, return type or parameters of the `SolveNinePuzzle` method. You may add additional methods as needed. The main method in the template contains code to help you test your implementation by entering test data or reading it from a file. You may modify the main method, but only the contents of the `SolveNinePuzzle` method (and any methods you have added) will be marked. Please read through the comments in the template file before starting.

2 Input Format

9-puzzle boards are input as 3×3 tables, with the character '0' representing the empty square. For example, the board



would be represented by the input sequence

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

3 Suggested Algorithm

The exact implementation of the `SolveNinePuzzle` method is up to you. However, the algorithm outlined below is suggested for simplicity:

- Construct the graph G of all states of the 9-puzzle.
- Find the vertex v of G corresponding to the input board B.
- Find the vertex u of G corresponding to the goal state.
- Run DFS or BFS on G starting at u.
- If v was encountered by the traversal, print each board on the v-u path and return `true`.
- Otherwise, return `false`.

In practice, puzzles like the 9-puzzle and 16-puzzle are solved with more advanced artificial intelligence algorithms, which are beyond the scope of this course.

Pseudocode for constructing G can be found in the additional notes in the resources tab of conneX. Each vertex of G corresponds to a possible 9-puzzle board, but it can be helpful to have vertices represented by integers instead of 3×3 arrays to facilitate indexing into an adjacency list structure. To enable this, you have been provided with two functions in the template:

- `getBoardFromIndex`(i): Given a board index i, return the corresponding board.
- `getIndexFromBoard`(B): Given a board B, return the corresponding index.

These functions will allow you to construct G as if vertices were integers. Since the number of vertices in G is large, you are encouraged to use an adjacency list representation for the graph.

If v is the vertex corresponding to the input board and u is the vertex corresponding to the goal state, then v is solvable if and only if a traversal (DFS or BFS) rooted at u encounters v. If v is never encountered, then `SolveNinePuzzle` should return `false`. If v is encountered, then the traversal tree computed by DFS or BFS can be used to find a path from u to v in the graph (specifically, by starting at v and tracing upward through the tree until u is reached). The implementation should print every board encountered along this path using the provided method `printBoard`(B). If your submission prints boards using any other means, it may lose marks.

4 Test Datasets

A collection of randomly generated solvable and unsolvable boards has been uploaded to conneX. Your assignment will be tested on boards similar but not identical to the uploaded boards. You may want to create your own collection of test boards.

5 Sample Run

The output of a model solution on the board specified above is given in the listing below. Note that there may be multiple move sequences that solve a given board. Console input is shown in blue.

```
Reading input values from stdin.
Reading board 1
4 1 3
0 2 6
7 5 8
Attempting to solve board 1...
4 1 3
0 2 6
7 5 8

0 1 3
4 2 6
7 5 8

1 0 3
4 2 6
7 5 8

1 2 3
4 0 6
7 5 8

1 2 3
4 5 6
7 0 8

1 2 3
4 5 6
7 8 0

Board 1: Solvable.
Processed 1 board.
Average Time (seconds): 0.00
```

6 Evaluation Criteria

The programming assignment will be marked out of 50, based on a combination of automated testing and human inspection. The running time of the implemented algorithm should be at most $O(n^2)$, where n is the number of vertices in the constructed graph G. To receive full marks, the implementation should solve each board with the smallest number of moves possible. This can be achieved by using BFS instead of DFS. Marks will be deducted if any method other than the provided `printBoard` function is used to print the sequence of moves needed to solve the board.

Score(/50)	**Description**
0 - 5	Submission does not compile or does not conform to the provided template.
5 - 25	The implemented algorithm is not $O(n^2)$ or is substantially inaccurate on the tested inputs.
26 - 40	The implemented algorithm is $O(n^2)$ and accurate on all tested inputs.
41 - 50	The implemented algorithm is $O(n^2)$, gives the correct answer on all tested inputs, and the sequence of moves for each tested board is the shortest possible length.

To be properly tested, every submission must compile correctly as submitted, and must be based on the provided template. **If your submission does not compile for any reason (even trivial mistakes like typos), or was not based on the template, it will receive at most 5 out of 50**. The best way to make sure your submission is correct is to download it from conneX after submitting and test it. You are not permitted to revise your submission after the due date, and late submissions will not be accepted, so you should ensure that you have submitted the correct version of your code before the due date. conneX will allow you to change your submission before the due date if you notice a mistake. After submitting your assignment, conneX will automatically send you a confirmation email. If you do not receive such an email, your submission was not received. If you have problems with the submission process, send an email to the instructor **before** the due date.