CS69001: Computing Lab – I
Autumn 2009

Assignment 3

Implementation of Backtracking

Due: August 25, 2009 (Tuesday)

In this exercise, you are required to write a backtracking algorithm for solving the 15 puzzle, or more correctly, the generalized $n^2 - 1$ puzzle. You start from an arbitrary arrangement of the tiles $1, 2, \ldots, n^2 - 1$ in an $n \times n$ board (this leaves exactly one blank square). Your task is to find a sequence of moves that restore the board to the following configuration (shown for $n = 4$), henceforth called the *final configuration*.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

For example, here is a sequence of moves:

1	2	3	4
5	6	8	
9	10	7	12
13	14	11	15

$\rightarrow$

1	2	3	4
5	6		8
9	10	7	12
13	14	11	15

$\rightarrow$

1	2	3	4
5	6	7	8
9	10		12
13	14	11	15

$\rightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14		15

$\rightarrow$

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Part 1 **(40)**

From exactly half of the initial configurations, it is possible to reach the final configuration. These *solvable* initial configurations are characterized as follows. First, write the initial configuration in the row-major order (neglecting the blank square). For example, the leftmost configuration in the above example is written as:

1	2	3	4	5	6	8	9	10	7	12	13	14	11	15

Let m denote the number of inversions in this array (that is, the number of pairs (i, j) such that $i > j$, but i appears earlier than j in the array). Also let n denote the width (or height) of the grid. Finally, let r be the index of the row in the 2-d grid (indexing starts from 0 at the top), containing the blank square. The initial configuration is solvable if and only if one of the following is true.

- If n is odd, then m is even.
- If n is even, then $m + r$ is odd.

For example, all the inversions in the above flattened array are $(8, 7)$, $(9, 7)$, $(10, 7)$, $(12, 11)$, $(13, 11)$ and $(14, 11)$, that is, $m = 6$. The row index of the blank square is $r = 1$. Since the grid size $n = 4$ is even, we use the second case. But $m + r = 7$ is odd, so the corresponding initial configuration is solvable. A sequence of steps leading to the final configuration is already shown above.

Write a function that, given a configuration of an $n \times n$ board, determines whether that configuration is solvable.

Part 2 **(60)**

Write a non-recursive function based upon backtracking in order to compute a sequence of moves that change the board from a given initial configuration to the final configuration (provided that the initial configuration is solvable). In order to avoid infinite loops in the search, you need to adopt two measures.

- No configuration is repeated on the search path.
- Possibilities of the movement of the blank tile are explored in a particular order.

You may implement a depth-restricted version of the backtracking procedure, and gradually increase the depth until a solution is reached. It is known that you may require as many as 31 (or 80) moves for $n = 3$ (or $n = 4$).

Submit a single C/C++ file solving both the above parts. The file must contain your name and roll number.