300103 Data Structures and Algorithms

Practical 3 (Week 4)

(2 Marks)

Task 3.1
Download the program under Task 3.1. The program can display any size of a rectangle board and take inputs from the user to fill the board. Add more code to the program so that it can record all the moves the user has made using a STL stack. Print out the content of the stack once the board is full.

Hint: *The implementation of the Board class is a minimum. Add as many more functions to the class as needed.*

Task 3.2
Based on the same program provided for Task 3.1, randomly generate moves until you fill the whole board. If the board size is *m*n*, you can only call the random function rand() *mn* times (excluding the random function rand() that have been used in the provided code).

Hint:
*1. For a m*n board, the coordinates of the k^{th} cell (start from the 0^{th}, called the index of the cell) can be calculated as follows:*

$i = k \% m$

$j = k / n$

For instance, if m=3 and n=5, the coordinates of the 8^{th} cell of the board are

$8 / 3 = 2$

$8 \% 5 = 3$

On the other side, given the coordinates of a cell, say (i, j), the index of the cell is

$i*m+j$

2. Use a vector of integers to store the indices of the empty cells.

Task 3.3
Based on the same program provided for Task 3.1, including Neighbour.h, add code into function `checkNeighbors(int x, int y)` to store and display the non-empty valid neighbors' information. For instance, if the current board is:

```
 --- --- ---
| 1 | 2 | 5 |
 --- --- ---
| 4 |   |   |
 --- --- ---
|   | 3 |   |
 --- --- ---
```

The (x, y) values are (1, 1), your program should display:

```
Up: 2
Down: 3
Left: 4
Total value: 9
```

Use either a vector or a list to store the information. Try to store objects of neighbours into the vector or list.