Practice Exercise #24: Valid Path

http://www.comp.nus.edu.sg/~cs1010/4_misc/practice.html

Reference: Week 6 Exercise #3

Date of release: 15 September 2014

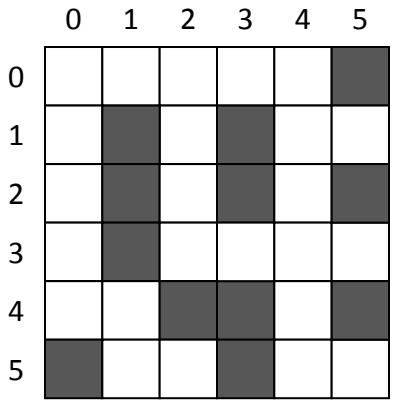
Objective: Two-dimensional array

Task statement:

A **maze** is represented by a two-dimensional 6×6 integer array, in which value 0 represents a wall and 1 represents a cell. The start and exit points in the maze are located at maze[0][0] and maze[5][5] respectively.

A **path** is represented by a character array. Each element is one of the 4 characters representing the directions: 'N' (north), 'S' (south), 'E' (east) or 'W' (west). The path is assumed to start at maze[0][0].

The figure on the left below illustrates an example of a maze, and the input data representing it is shown on the right.




```
1 1 1 1 1 0
1 0 1 0 1 1
1 0 1 0 1 0
1 0 1 1 1 1
1 1 0 0 1 0
0 1 1 0 1 1
```

A path in a maze is defined to be valid if the path is within the maze and does not knock against any wall. For the given maze above, "EESNEES" is a valid path, while "SSW" and "SSSE" are not valid. You may assume that a path contains at most 10 characters.

Write a program **validPath.c** to check whether a given path is valid in a given 6×6 maze.

The skeleton program provided contains most of the code. You are to the complete the **isValid ()** function, which returns 1 if the path is valid, or 0 otherwise.

Sample runs:

```
Enter maze:
1 1 1 1 1 0
1 0 1 0 1 1
1 0 1 0 1 0
1 0 1 1 1 1
1 1 0 0 1 0
0 1 1 0 1 1
Enter path: EESNEES
Path is valid.

Enter maze:
1 1 0 0 1 0
1 0 1 0 1 1
1 1 1 1 1 0
0 0 0 1 0 1
1 1 1 1 1 0
0 1 1 0 1 1
Enter path: EWSSENES
Path is not valid.
```