Assignment 1: Maze (due 11/27)

Write a program that finds out a path out in a maze. A 4*7 maze can be represented as follows:

```
4 7
1011110
1011000
1000011
0010110
```

Here, '0' represents that there is no obstacle (and '1' otherwise), so one can only visit the '0'-locations. The bottom-left and top-right locations are always '0', indicating there is always an entrance and an exit in the maze. Your program needs to find a way from the **bottom-left**, represented as **(1,1)**, to the **top-right** and output the results as follows:

```
(1,1)->(2,1)->(2,2)->(3,2)->(4,2)->(5,2)->(5,3)->(6,3)->(7,3)->(7,4)
```

If there is no way to the destination, output

```
No way
```

What You Need to Do

Read in a m×n array representing a maze. You do not have to check input's validity. Your program should output the path from the entrance (bottom-left) to the exit (top-right). *You will get extra bonus if your program can deal with multiple paths to the destination (if they exist) and output them in the following order (separated by '\n')*: at the point when they branch out, always go *right* first, followed by *up*, *left*, *down*. If no such path exists, your program should output "No way\n". **Do not use static arrays or variable-length arrays. Use malloc**. You will lose penalty points if you use static or variable-length arrays.

Sample Input

```
4 7
1011110
1011000
1000011
0010110
```

Sample Output

```
(1,1)->(2,1)->(2,2)->(3,2)->(4,2)->(5,2)->(5,3)->(6,3)->(7,3)->(7,4)
```

Hint

You may need to use a **stack** to record your previous moves to the current location and another mxn array to record the locations you have visited (to prevent from looping forever). This is probably the easiest way. Alternatively, you can use recursion without using a stack, since recursion automatically uses the system stack. You may still use recursion, but we encurage you to use **iteration** instead of recursion, since recursion is actually **harder** in this case. There may be other ways to do this assignment, but they may be harder.

Grading Criteria

Correctness 70%

This will be checked by OJ just like our Lectures and Labs.

Program Style 30%

1. Your program should be **well-commented**. All variables, functions, loops should be clearly explained both in the source code and briefly in the README file.
2. Blocks in your source code should have proper indentation (縮排).
3. Do not use any global variable to pass data in and out of a function.
4. For a good program style, please follow Recommended C style and coding rules.

Submission

1. Submit your source code on **OJ**.
2. Submit a README file on **eLearn** that briefly explains how you implement your work.