Cosc 1p03 Assignment 4

(Due date Mar 26th 16:00 est, Late date Mar 29th 16:00 est)

BackGround

Hansel has gotten himself lost in the forest and is often the case Gretel must air drop into the forest to locate him.

Objective

In this assignment you will be exploring a recursive solution to the maze walk problem.

Maze

You are given a maze which will serve as the forest as an ASCIIDatafile. This maze has no exits. It is expected that once Gretel locates Hansel they will magically teleport out. All walls of the maze are represented as '#'. The size of the maze (rows, col) is given as an integer pair i.e. (8,11). Below is a sample maze.

```
8    11
###########
#         #
#### ######
#    #    #
#    #    #
# ###### ##
#         #
###########
```

Once, you run your maze walk application a path is traced from the starting location G (in this case (1,1)) to Hansel marked with an H, using ASCII characters. The starting location will be randomly generated.

```
###########
#G>>v     #
####v######
# v<v#  >H#
#v<^<#  ^.#
#v######^##
#>>>>>>>^ #
###########
```

The Maze Walk Algorithm

The recursive maze walk is similar to the image scan provided in class. The premise is to try a direction and if successful keep trying that direction until the goal condition is met or a wall is encountered. This is repeated for all four directions. Note: only 4 directions are considered. If a location has been exhausted, i.e. all four directions, then it is marked with a "." so that no further attempts are made at that location. This prevents infinite recursive calls. Consider the case where a location can be entered from multiple directions. The solution exemplified above shows one such case, and is dependent on order of the recursive calls.

Write a method findPath(int x, int y) which is recursively called in order to find a path through the maze finding Hansel. You may modify findPath as appropriate to include additional parameters or a return value to help facilitate the path tracing as in the example.

Hansel (H) and Gretel (G) are to be randomly placed in the maze. You may generate a random row and column within the bounds of the maze. If the location is not a wall accept the location and place H and then repeat and place G.

Output should be the starting location **G**; maze with both H and G indicated followed by where H was found. Include the path through the maze using ASCII ^v<> characters. After the maze has been printed output the row and column where Hansel has been found. E.g. of output expected

```
Starting location G is (x,y)

###########
#G>>v     #
####v######
# v<v#  >H#
#v<^<#  ^.#
#v######^##
#>>>>>>>^ #
###########

Hansel found at location (x,y)
```

Little Hints

1) When reading characters from the ASCIIDataFile use the `readC()` method. This method reads exactly one ASCII character.
2) At times you may have extra spaces at the end of some lines of text or simply wish to ignore the rest of the characters on the current line since hidden characters like line feeds are still characters but not part of the matrix (forest). The method `readLine()` will read all characters from the current location to the end of line including the line feed. Allowing the next read to start at the beginning of the next line.
3) Included are 2 maze files mz1.txt and mz2.txt. These can be used as input maze maps.

4) Use an ASCIIOutputFile. Completed solutions should look similar to the sample given in this text. When viewing the file use a mono space font like courier so that the maze appears properly.
5) Depending on the order in which you explore unvisited locations in the map, the final solution may vary slightly indicating a different paths or tried locations.
6) The algorithm will find a solution, not necessarily the optimal solution.
7) As with most recursive algorithms the solution will be very few lines of code. If findPath seems to be getting very complicated and big, then chances are you are doing it wrong.
8) Note: There are multiple base cases, **H**, ‘**.**’ (which indicates all possible solutions have been tried), and a wall #.

Submission

Submit a zip file of your assignment via Sakai. Include with your submission a comprehensive output solution for the large maze. Be sure to provide proper commenting and layout of your code.