Homework 4: Due: Tuesday March 27 by 11:59PM via email to grader  
(dkc237@nyu.edu)

This homework assignment is to write an applet to find the solution to a maze, if one exists, or to print a message that no path was found. Here are the details.

A maze is a two dimensional array with some number of rows and columns (for our purposes use 19 rows and 21 columns please). An entry in the maze is "blocked" with some probability (I use cutoff = .35, i.e. if a random number between 0 and 1 is less than the cutoff that the cell is blocked; otherwise it is open). A path can start in any open cell in the leftmost column 0 (any row), and ends in the rightmost column (i.e. column-1). You are allowed to move in any of the 4 directions (N,E,S,W) in the maze if it is open, and doesn’t lead to an index out of bounds.

You should use a recursive approach to searching the maze. This will not necessarily lead to the shortest path, but if there is one you will find it. Note that you will have to mark the cells in the maze that you have already visited, so you don’t revisit them in an infinite loop. Also, besides marking them internally you should draw it on your applet, that is, for each cell that you visit, you should mark it with an “X”. In addition, as demonstrated in class, you should mark the cells that you backtracked from in a different color, so you can see what your program is doing.

Remember, as discussed in class, you will need to use a separate thread with a start routine and a run method for the drawing on the applet frame to happen step by step instead of waiting until the end of your program. This happens even if you call repaint(). Also, you will have to slow things down (using Thread.sleep) to see what your program does at each step. See the applet from class for an example.

Honors Section: You should write a second program that finds the shortest path through the maze, and draws that on the applet. Here is the outline of an algorithm to do this, using a queue data structure. Put all cells where you can enter the maze in the queue, and mark them with a 1. All of their neighbors that are not yet marked with a smaller number and are not BLOCKED then have a path length of two. You continue this (dequeuing the path length n cells, marking their accessible neighbors with n+1 and enqueueing them) until the exit column is reached. For this program as well as the previous one your applet should number the cells with the path length to reach them and draw it.

Extra credit (Honors section): once you have found the shortest path length, mark it in a different color so you can see which path through the maze was taken.