# MazeGrandMaster 2



MazeMaster returned booleans to mark a path, any path, through the maze.

MazeGrandMaster 1 returned ints to calculate the shortest path. It did not mark the path.

MazeGrandMaster 2 will return Strings to build and then to calculate the shortest path. You will implement three methods:

/\*\*recur until you find E, then build the path with (r,c) locations   
 and the number of steps, e.g. ((5,0),10),((5,1),9),((6,1),8),

((6,2),7),((6,3),6),((6,4),5),((6,5),4),((6,6),3),((5,6),2),

((4,6),1),((4,7),0)  
 as you build, choose the shortest path at each step  
 returns empty String if there is no path  
 precondition: Start can't match with Exit

\*/

public String findShortestPath(int r, int c)

{}

/\*\*returns the length, i.e., the third number when the format of

strPath is "((3,4),10),((3,5),9),..."  
 returns 999 if the string is empty  
 precondition: strPath is either empty or follows the format above

\*/  
public int getPathLength(String strPath)

{}

/\*\* recursive method that takes a String created by findShortestPath(r, c)   
 in the form of ((5,0),10),((5,1),9),((6,1),8),((6,2),7),

((6,3),6),((6,4),5),((6,5),4),((6,6),3),((5,6),2),((4,6),1),

((4,7),0) and marks the actual path in the maze  
 precondition: the String is either an empty String or one that

has the format shown above  
 the (r,c) must be correct for this method to work

\*/

public void markPath(String strPath)

{}



**Extension**: above we used a large arbitrary number 999, which is not a good programming practice. We really ought to use -1 to indicate when the path fails. Do that.