COS101

Practical 6

24-26 October 2018

Theme: Recursion

For this week's practical an incomplete template for solving a maze using recursion has been provided.

Your task is,

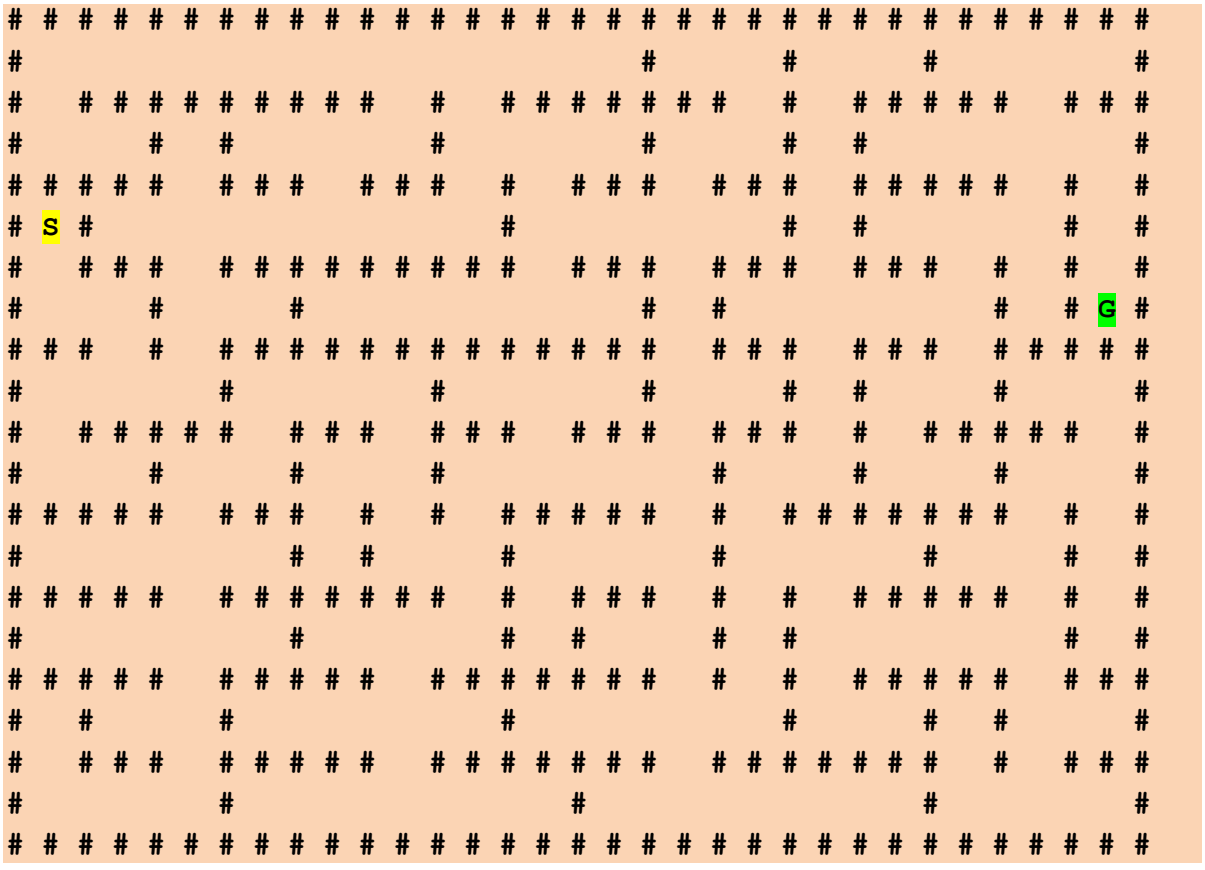
(1) to convert the python code to Java and

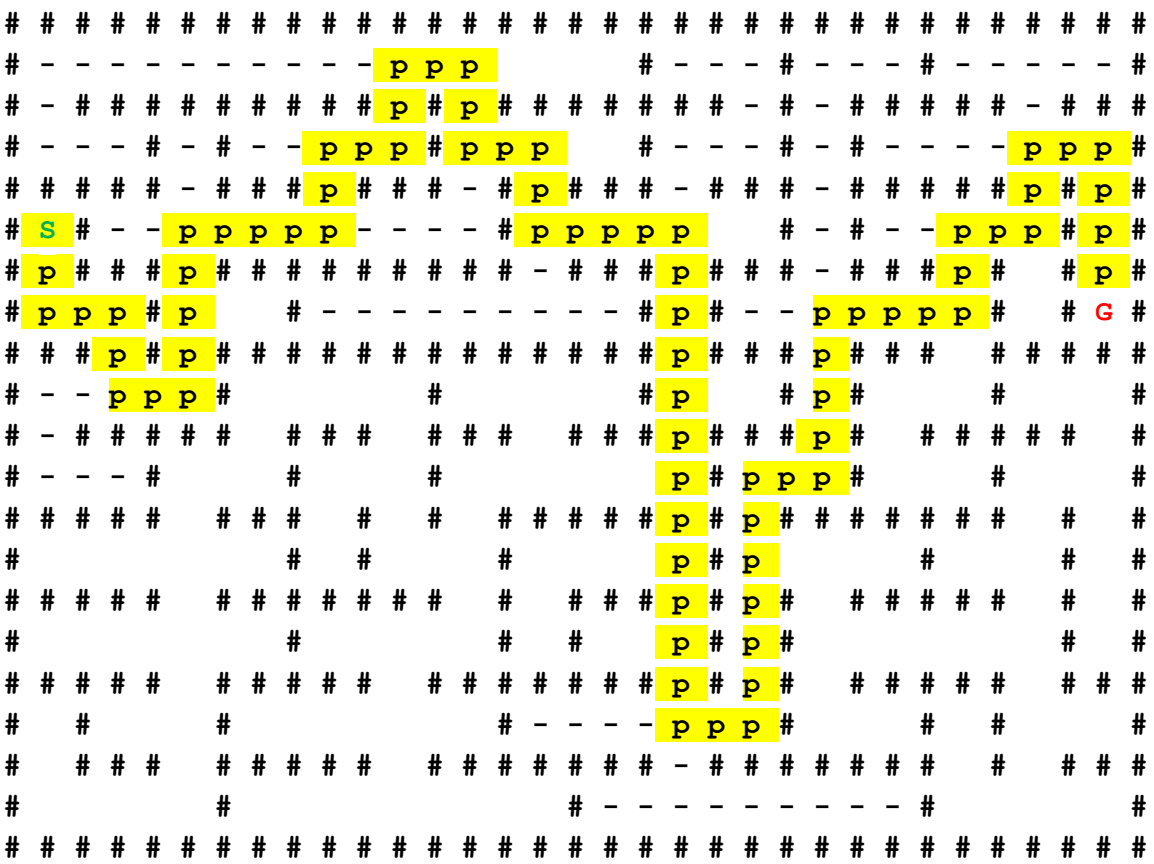
(2) to provide the recursive part in the yellow shaded text.

Use the maze indicated below when solving the problem. Note how the maze is stored in the program.

Sample output for the program appears on page 2. The path traversed is shaded yellow.

Save your program as **SolveMazeRecursively.java**.





Maze solved!

```
#
# This program solves a two dimensional maze using recursion
# ----------------------------------------------------------

endRow = 7      # row of goal position
endCol = 31     # column of goal position

startRow = 5    # row of starting position
startCol = 1    # column of starting position

# The maze contains the following rows
# ------------------------------------

r1 = [ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1 ]
r2 = [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
    0, 0, 1, 0, 0, 0, 0, 0, 1 ]
r3 = [ 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0,
    1, 1, 1, 1, 1, 0, 1, 1, 1 ]
r4 = [ 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 0, 0, 0, 0, 1 ]
r5 = [ 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
    1, 1, 1, 1, 1, 0, 1, 0, 1 ]
r6 = [ 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 0, 0, 1, 0, 1 ]
r7 = [ 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
    1, 1, 1, 0, 1, 0, 1, 0, 1 ]
r8 = [ 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
    0, 0, 0, 0, 1, 0, 1, 0, 1 ]
r9 = [ 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
    1, 1, 1, 0, 1, 1, 1, 1, 1 ]
r10 =[ 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 1 ]
r11 =[ 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
    1, 0, 1, 1, 1, 1, 1, 0, 1 ]
r12 =[ 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
    1, 0, 0, 0, 1, 0, 0, 0, 1 ]
r13 =[ 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1,
    1, 1, 1, 1, 1, 0, 1, 0, 1 ]
r14 =[ 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
    0, 0, 1, 0, 0, 0, 1, 0, 1 ]
r15 =[ 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
    1, 1, 1, 1, 1, 0, 1, 0, 1 ]
r16 =[ 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0,
    0, 0, 0, 0, 0, 0, 1, 0, 1 ]
r17 =[ 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0,
    1, 1, 1, 1, 1, 0, 1, 1, 1 ]
r18 =[ 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0,
    0, 0, 1, 0, 1, 0, 0, 0, 1 ]
r19 =[ 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
    1, 1, 1, 0, 1, 0, 1, 1, 1 ]
r20 =[ 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 1, 0, 0, 0, 0, 0, 1 ]
r21 =[ 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 1 ]

maze                                                                    =
    [r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13,r14,r15,r16,r17,r18,r19,r20,r21
    ]
```

```
pathFound = False

def buildMaze(Amaze):
    maze = Amaze


def mark(Amaze, row, col):
    Amaze[row][col] = 2

def unMark(Amaze, row, col):
    Amaze[row][col] = -2

def isMarked(Amaze, row, col):
    if Amaze[row][col] == 2:
        return True
    else:
        return False


def openRight(Amaze, row, col):
    return Amaze[row][col+1] == 0

def openLeft(Amaze, row, col):
    return Amaze[row][col-1] == 0

def openTop(Amaze, row, col):
    return Amaze[row-1][col] == 0

def openBottom(Amaze, row, col):
    return Amaze[row+1][col] == 0



def getStartingRow():
    return startRow

def getStartingColumn():
    return startCol

def isGoal(Amaze, row, col):
    return (row == endRow) and (col == endCol)


def nextMove(dir):
    if dir == "UP":
        return "DOWN"
    elif dir == "DOWN":
        return "LEFT"
    else:
        if dir == "LEFT":
            return "RIGHT"
```

```
def markPath(r, c, Amaze):
    if isGoal(Amaze, r, c):
        found = True
    else:
        if isMarked(Amaze, r, c):
            found = False
        else:
            mark(Amaze, r, c)
            found = False
            dir = "UP"
            done = False
            while not done:
                if dir == "UP":
                    // insert your code here
                elif dir == "DOWN":
                    // insert your code here
                elif dir == "LEFT":
                    // inset your code here
                else:
                    if dir == "RIGHT":
                        // insert your code here
                if found or (dir == "RIGHT"):
                    done = True
                if not done:
                    dir = nextMove(dir)
           if not found:
                unMark(Amaze, r, c)
    return found

def solveMaze():
    global pathFound, maze
    buildMaze(maze)
    printMaze(maze)
    startR = getStartingRow()
    startC = getStartingColumn()
    pathFound = markPath(startR, startC, maze)
    if pathFound:
        printMaze(maze)
        print "Maze solved!"
    else:
        print "No path found"
```

```
def character(r,c):       #  appropriate  characters  provided  for  printing
    purposes
    arg = maze[r][c]
    if r == startRow and c == startCol:
        return "S"       #  indicates starting position in maze
    elif r == endRow and c == endCol:
        return "G"
    elif arg == 0:        #  indicates starting position in maze
        return " "        # indicates unvisited cell
    elif arg == 1:
        return "#"        # indicates wall
    elif arg == 2:        # indicates cell visited when solving maze
        return "p"
    else:
        if arg == -2:     # indicates cell visited but unmarked when it was not
    leading to a solution
            return "-"

def printMaze(Amaze):     # print the entire maze
    for r in range(len(Amaze)):
        for c in range(len(Amaze[0])):
            print character(r,c),
        print
    print
    print

# Start of main routine
# ----------------------

solveMaze()
```