```
public static boolean ratInAMaze(int maze[][]){
        int n = maze.length;
        int path[][] = new int[n][n];
        return solveMaze(maze, 0, 0, path);
}

public static boolean solveMaze(int maze[][], int i, int j, int path[][]){
        int n = maze.length;
        // Check if i,j cell is valid or not
        if(i  < 0 || i >= n || j < 0 || j >= n  || maze[i][j] == 0
                ||          path[i][j] == 1){
                return false;
        }
        // Include the cell in current path
        path[i][j] = 1;
        // Destination cell
        if(i == n - 1 && j == n - 1){
           for(r = 0;r<n;r++){
              for(int c=0;c<n;c++) {
                 System.out.print(path[r][c] + "")
              }
               System.out.println();
           }
           return true;
                path[i][j] = 0;
                return true;
        }

        // Explore further in all directions
        boolean pathPossible = false;
        // top
        if(solveMaze(maze, i - 1, j, path)){
                pathPossible = true;
        }
        // right
        else if(solveMaze(maze, i, j + 1, path)){
                pathPossible = true;
        }
        // Down
        else if(solveMaze(maze, i + 1 , j, path)){
                pathPossible = true;
        }
        // Left
        else if(solveMaze(maze, i, j - 1, path)){
                pathPossible = true;
        }
```

```
                return pathPossible;
        }

}
```