



In []:

```

N = 4

def printSolution( sol ):
    for i in sol:
        for j in i:
            print(str(j) + " ", end = "")
        print("")

def isSafe( maze, x, y ):

    if x >= 0 and x < N and y >= 0 and y < N and maze[x][y] == 1:
        return True

    return False

def solveMaze( maze ):

    # Creating a 4 * 4 2-D List
    sol = [ [ 0 for j in range(4) ] for i in range(4) ]

    if solveMazeUtil(maze, 0, 0, sol) == False:
        print("Solution doesn't exist");
        return False

    printSolution(sol)
    return True

# A recursive utility function to solve Maze problem
def solveMazeUtil(maze, x, y, sol):

    # if (x, y is goal) return True
    if x == N - 1 and y == N - 1:
        sol[x][y] = 1
        return True

    # Check if maze[x][y] is valid
    if isSafe(maze, x, y) == True:
        # mark x, y as part of solution path
        sol[x][y] = 1

        # Move forward in x direction
        if solveMazeUtil(maze, x + 1, y, sol) == True:
            return True

        # If moving in x direction doesn't give solution
        # then Move down in y direction
        if solveMazeUtil(maze, x, y + 1, sol) == True:
            return True

        # If none of the above movements work then
        # BACKTRACK: unmark x, y as part of solution path
        sol[x][y] = 0
        return False

maze = [ [1, 0, 0, 0],
          [1, 1, 0, 1],
          [0, 1, 0, 0],

```

```
[1, 1, 1, 1] ]
```

```
solveMaze(maze)
```