1. From the rat in a maze problem, explain the code

```
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
```

This method is written for solve the solution if there no solution at the first place [matrix row 0 colum 0] there will say that “Solution doesn’t exist” otherwise it will print how the rat goes though the maze

```
# 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 and maze[x][y]== 1:
        sol[x][y] = 1
        return True

    # Check if maze[x][y] is valid
    if isSafe(maze, x, y) == True:
        # Check if the current block is already part of solution path.
        if sol[x][y] == 1:
            return False

        # mark x, y as part of solution path
        sol[x][y] = 1

        #force rat to go to the right way
        if solveMazeUtil(maze, x + 1, y, sol):
            return True

        if solveMazeUtil(maze, x, y + 1, sol):
            return True

        if solveMazeUtil(maze, x - 1, y, sol):
            return True

        if solveMazeUtil(maze, x, y - 1, sol):
            return True

        sol[x][y] = 0
        return False
```

This method is guide the rat though the maze (we will talk about the is safe next session). The method will check all direction of the path it if rat can go to the path it will force the rat to go instantly

```python
# A utility function to check if x, y is valid
# index for N * N Maze
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
```

This method is checking the current path that rat will goes can go or not by checking if there are 1 (mean can go) or is it a bound of the maze or not if rat can go it will response true otherwise false

```python
# A utility function to print solution matrix sol
def printSolution( sol ):
    for i in sol:
        for j in i:
            print(str(j) + " ", end ="")
        print("")
```

This method will call when the rat goes out of the maze or there are no way to go because of we are represented the maze as a matrix so we use 2 for loop to print a path of the maze

```python
# Driver program to test above function
if __name__ == "__main__":
    # Initialising the maze
    maze = [ [1, 0, 0, 0],
             [1, 1, 0, 1],
             [0, 1, 0, 0],
             [1, 1, 1, 1] ]

    solveMaze(maze)
```

This is a driver method that will give the maze and let the rat goes

2. Show each step of code when the maze is step 1

Color term:
red: current path
Blue: path check
green: passed path

Step	Real Maze	Rat path
1	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
2	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
3	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0

4	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0
5	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 1 0 0
6	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0
7	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1
8	1 0 0 0 1 1 0 1 0 1 0 0 1 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1

Optional
Windows 10 education
Visual studio code January update
python 3.8
code from Shiv Shankar

Explain the code and visualize by Sahachan Tippimwong
submit to Aj Algorithm