S.No: 1	Exp. Name: ***Project Module***	Date: 2024-05-22

Aim:

Project Module.

Source Code:

CTP28132.py

```
def print_maze(maze):
    for row in maze:
        print(' '.join(str(cell) for cell in row))
    print()

def is_valid_move(maze, visited, x, y):
    return 0 <= x < len(maze) and 0 <= y < len(maze[0]) and not visited[x][y] and maze[x][y]
== 0

def dfs(maze, x, y, visited, path):
    if x == len(maze) - 1 and y == len(maze[0]) - 1:  # goal condition (bottom-right corner)
        path.append((x, y))
        return True

    if is_valid_move(maze, visited, x, y):
        visited[x][y] = True
        path.append((x, y))

        # Explore all four directions: down, right, up, left
        if dfs(maze, x + 1, y, visited, path):  # down
            return True
        if dfs(maze, x, y + 1, visited, path):  # right
            return True
        if dfs(maze, x - 1, y, visited, path):  # up
            return True
        if dfs(maze, x, y - 1, visited, path):  # left
            return True

        # Backtrack if none of the moves work
        path.pop()
        visited[x][y] = False

    return False

def solve_maze(maze):
    rows, cols = len(maze), len(maze[0])
    visited = [[False for _ in range(cols)] for _ in range(rows)]
    path = []

    if dfs(maze, 0, 0, visited, path):
        return path
    else:
        return "No solution"

def visualize_solution(maze, path):
    maze_with_path = [row[:] for row in maze]
    for x, y in path:
        maze_with_path[x][y] = '*'
    print_maze(maze_with_path)

def main():
    # Define the maze here, 0 is path, 1 is wall
    maze = [
        [0, 1, 0, 0, 0],
        [0, 1, 0, 1, 0],
```

```
        [0, 0, 0, 0, 0]
    ]

    print("Initial Maze:")
    print_maze(maze)

    solution = solve_maze(maze)

    if solution != "No solution":
        print("Path to the solution:")
        for step in solution:
            print(step)
        print()

        print("Maze with solution path:")
        visualize_solution(maze, solution)
    else:
        print("No solution found for the maze.")

def create_maze_from_input():
    print("Enter the dimensions of the maze (rows and columns):")
    rows = int(input("Rows: "))
    cols = int(input("Columns: "))

    maze = []
    print("Enter the maze row by row (0 for path, 1 for wall):")
    for i in range(rows):
        row = list(map(int, input().split()))
        if len(row) != cols:
            print(f"Row {i + 1} does not have {cols} columns. Please re-enter this row.")
            row = list(map(int, input().split()))
        maze.append(row)

    return maze

if __name__ == "__main__":
    choice = input("Do you want to enter a custom maze? (yes/no): ").strip().lower()
    if choice == 'yes':
        maze = create_maze_from_input()
    else:
        # Default maze if no custom maze is entered
        maze = [
            [0, 1, 0, 0, 0],
            [0, 1, 0, 1, 0],
            [0, 0, 0, 1, 0],
            [0, 1, 1, 1, 0],
            [0, 0, 0, 0, 0]
        ]

    print("Initial Maze:")
    print_maze(maze)

    solution = solve_maze(maze)

    if solution != "No solution":
```

```
            print(step)
        print()

        print("Maze with solution path:")
        visualize_solution(maze, solution)
    else:
        print("No solution found for the maze.")
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Hello World
Hello World

ID: 2303811710422167
2023-2027-K
K.Ramakrishnan College of Technology