E:\maze.c

```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>

const char direction[] = "DLRU";
const int dr[] = {1, 0, 0, -1};
const int dc[] = {0, -1, 1, 0};

const int n = 4; // Define n as the size of the maze

bool isValid(int r, int c, int maze[n][n]) {
    return r >= 0 && c >= 0 && r < n && c < n && maze[r];
}

void solve(int r, int c, int maze[n][n], char currentPath[], char ans[][100], int* count) {
    if (r == n - 1 && c == n - 1) {
        strcpy(ans[*count], currentPath);
        (*count)++;
        return;
    }

    maze[r] = 0;

    for (int i = 0; i < 4; i++) {
        int nextr = r + dr[i];
        int nextc = c + dc[i];

        if (isValid(nextr, nextc, maze)) {
            currentPath[strlen(currentPath)] = direction[i];
            solve(nextr, nextc, maze, currentPath, ans, count);
            currentPath[strlen(currentPath) - 1] = '\0';
        }
    }

    maze[r] = 1;
}

void findPath(int maze[n][n]) {
    if (maze[0][0] == 1) {
        char currentPath[100];
        char ans[100][100];
        int count = 0;
        currentPath[0] = '\0';
        solve(0, 0, maze, currentPath, ans, &count);

        if (count == 0)
            printf("-1");
        else {
            for (int i = 0; i < count; i++) {
                printf("%s ", ans[i]);
            }
        }
    }
}
```

```c

int main() {
    int maze[4][4] = { {1, 0, 0, 0},
                       {1, 1, 0, 1},
                       {1, 1, 0, 0},
                       {0, 1, 1, 1} };

    findPath(maze);

    return 0;
}

```