

```

1  //////////////////////////////////////
2  //
3  // Author      : Gregorio Lozada
4  // Date        : 4/5/2018
5  //
6  // Homework Assignment 3:  Maze
7  //
8  // This program solves an 8 x 13 maze.
9  //
10 // Output      : Maze displaying the solved patth and dead ends.
11 //
12 //////////////////////////////////////
13
14 #include <stdio.h>
15 #include "Stack.h"
16
17 void MovePoint(Stack *stack, Point *runner, int row, int column);
18
19 int main() {
20     //Variables
21     char maze[10][16] =
22     { {"00000000000000"},
23       {"0 0 0 000 0 00"},
24       {"0          0"},
25       {"0 0 0000000 0 0"},
26       {"0 0  0  0  0"},
27       {"0 000 00 0000"},
28       {"0  0 0  0"},
29       {"00 0 0 0 000 00"},
30       {"00  0  0  0"},
31       {"000000000000000"}
32     };
33
34     int mazeSize = (sizeof(maze) / sizeof(maze[0][0]));
35
36     Stack stack(mazeSize);
37
38     Point runner(1, 1);
39     Point end(8, 13);
40
41     //WHILE runner is not at the end of maze and stack is not empty
42     do {
43         /*
44         IF there's an empty space above, below, or next to runner
45             Change maze character to '*'
46             MovePoint
47         ELSE
48             Change maze character to 'D'
49             Set runner point to popped point in stack

```

```
50     */
51     if (maze[runner.row][runner.column + 1] == ' ') {
52         maze[runner.row][runner.column] = '*';
53         MovePoint(&stack, &runner, runner.row, runner.column + 1);
54     }
55     else if (maze[runner.row - 1][runner.column] == ' ') {
56         maze[runner.row][runner.column] = '*';
57         MovePoint(&stack, &runner, runner.row - 1, runner.column);
58     }
59
60     else if (maze[runner.row][runner.column - 1] == ' ') {
61         maze[runner.row][runner.column] = '*';
62         MovePoint(&stack, &runner, runner.row, runner.column - 1);
63     }
64     else if (maze[runner.row + 1][runner.column] == ' ') {
65         maze[runner.row][runner.column] = '*';
66         MovePoint(&stack, &runner, runner.row + 1, runner.column);
67     }
68     else {
69         maze[runner.row][runner.column] = 'D';
70         runner.Set(stack.Pop());
71     }
72 }
73 while ((runner.row != end.row || runner.column != end.column)
74        && stack.Size() != 0);
75
76 /*
77 IF runner made it our of the maze successfully
78     PRINT congratulations message
79 ELSE
80     PRINT "Cannot solve maze."
81 */
82 if (stack.Size() != 0) {
83     maze[end.row][end.column] = '*';
84
85     printf("Congratulations!\n");
86     printf("You made it out of the maze!\n");
87 }
88 else {
89     printf("Cannot solve maze.\n");
90 }
91
92 printf("\n");
93
94 //PRINT maze with path
95 for (int i = 0; i < (sizeof(maze) / sizeof(maze[0])); i++) {
96     for (int j = 0; j < (sizeof(maze[0]) / sizeof(maze[0][0])); j++) {
97         printf("%c", maze[i][j]);
98     }
```

```
99     printf("\n");
100 }
101
102     printf("\n");
103
104     return 0;
105 }
106
107 void MovePoint(Stack *stack, Point *runner, int row, int column) {
108     //PUSH runner point to stack
109     stack->Push(Point(runner->row, runner->column));
110
111     //SET runner point to empty point
112     runner->Set(row, column);
113 }
```