Birla Institute of Technology & Science, Pilani

Data Structures & Algorithms (CS F211)

Lab – 5 (Maze Search)

Problem

Brief:

- **Implement an iterative version of the Maze problem described below**

Maze Problem:

Given a Maze, it desired to find a path from a given *starting position* to the location with cheese.(*target position*). Maze is visualized as a rectangular grid containing walls (represented as 1) or free ways (represented as 0). The program must print a path that starts from starting location to the ending location. The target location will have value 9 stored in it. In this program, it is assumed that cyclic paths are not possible in the grid. (You need not check for cyclic paths.)

Algorithm:
A rough recursive algorithm is described below:

```
boolean traverse (Maze, posi, posj, &path_so_far, past_i, past_j)
{
if (posi,posj) is cheese,
     set path_so_far = { (posi,posj) }
     return true.
else if path to up is free, and we didn't come to this position from up
     ret=traverse (Maze, posi-1, posj, &path_so_far, past_i, past_j)
           if ret==true
               set path_so_far = { (posi,posj) ,  path_so_far } and return true
else if path to down is free and we didn't come to this position from down
     ret=traverse (Maze, posi+1, posj, &path_so_far, past_i, past_j)
           if ret==true
               set path_so_far = { (posi,posj) ,  path_so_far } and return true
else if path to left is free and we didn't come to this position from left
     ret=traverse (Maze, posi, posj-1, &path_so_far, past_i, past_j)
           if ret==true
               set path_so_far = { (posi,posj) ,  path_so_far } and return true
else if path to right is free and we didn't come to this position from right
     ret=traverse (Maze, posi, posj+1, &path_so_far, past_i, past_j)
     if ret==true
               set path_so_far = { (posi,posj) ,  path_so_far } and return true

return false  // case where we encountered dead end. All traverses returned false.
}
```

[Note that in all the set path_so_far statements, the current position is prepended to the path so far. End Note]

Description:

The program must take as input, a file containing the Maze, say maze.txt; and two other command-line arguments to indicate the starting coordinates(startx, starty). The program must then call the findCheese function that returns the path to the cheese from the starting position. The path must be printed in an output file output.txt.

Type definitions: (**`Maze.h`**)

(a) Create a type definition for the type **`Maze`**, which is an SIZE X SIZE grid of **`ints`**, which contain either 0 or 1 to indicate free way or wall respectively. The location with cheese will have 9 stored in it.

(b) Create a type definition for the enumerated type **`Boolean,`** that may be **`TRUE`** or **`FALSE.`**

Header file for operations: (**`MazeOps.h`**)

(c) **`Boolean findCheese(Maze m, int n, int posi, int posj, char **path_so_far, int past_i, int past_j)`**: Implementation of the above algorithm. Note that Maze m is a grid of size n x n. **`posi`** and **`posj`** are the row and column values of the location to try next. **`past_i`** and **`past_j`** are the row and column values of the previous position of the call. For the initial call, the values of **`past_i`** and **`past_j`** shall be -10.

(d) **`int createMaze(Maze *pm, char *mazefilename)`**: This function takes the name of the file containing the grid, **`mazefilename`**. It returns the size of the Maze created. (Implementation in createMaze.c)

Step 1:

Create findcheese_iter.c, which contains the iterative version of the algorithm given above. The file findcheese_rec.c provided contains the recursive version. This program takes the following as command-line parameters: name of file containing maze,

Step 2: driver file.

Create a driver file (**`maze.c`**) for meeting the objectives of this problem. This creates the Maze using **`createMaze`**. It then calls **`findCheese`** to determine the path. The path is printed onto to output.txt. The driver will take name of file containing maze, and name of output file as input parameters. (Default name of outputfile is output.txt)

The file format for output file is indicated below:

(x1,y2), (x2,y2), …

Deliverables: findcheese_iter.c, maze.c, make file, output file (say, output.txt), executable (maze), Maze.h, MazeOps.h, findcheese_rec.c, createMaze.c