Rat in a Maze Report

Algorithm

```
Procedure rat_in_maze(MAZE,STACK,TOP,DIR,SOL)
//MAZE(8,13)- 8x13 maze having ‘#’ as walls, ‘ ‘ as empty cell, and ‘O’ as rat initially at (0,0)
//STACK(104,2)- stack to store cell coordinates visited by rat
//TOP- points to location of topmost element in STACK
//DIR(8,2)- array containing all 8 direction coordinates for rat to move
//SOL(8,13)- solution path to exit out of maze. Initially contains only 0s
[
         STACK(0,0), STACK(0,1) ← 0,0;    TOP←0;
        SOL(0,0)←1;    row, col←0,0;
        DIR(0,0), DIR(0,1)←0,1;    DIR(1,0), DIR(1,1)←1,0;    DIR(2,0), DIR(2,1)←1,1;
        DIR(3,0), DIR(3,1)←1,-1;    DIR(4,0), DIR(4,1)← -1,0;    DIR(5,0), DIR(5,1)← -1,1;
        DIR(6,0), DIR(6,1)← -1,-1;    DIR(7,0), DIR(7,1)← 0,-1;

        if(MAZE(7,12)==’#’)
        [
                 print ‘NO SOLUTION’;
                return FALSE;
        ]
        while(TOP!=-1)
        [
                d←0;
                if(MAZE(7,12)==’O’)
                [
                        print ‘SOLUTION FOUND. SOLUTION PATH IS:- ‘;
```

```
          for i←0 to TOP
          [
                    SOL(STACK(i,0), STACK(i,1))←1;      print  STACK(i,0),
          STACK(i,1);
          ]
          for i←0 to 7
          [
                    print ‘\n’;
                    for j←0 to 12
                    [
                              print  SOL(i,j);
                    ]
          ]
return TRUE;
]//end if

while(d<8)
[
          if(valid(row+DIR(d,0),col+DIR(d,1)))
          [
                   MAZE(row,col)←'x'; pr←row; pc←col;
                    row← row+DIR(d,0); col←col+DIR(d,1);
          MAZE(row,col)←'O’;
                    TOP←TOP+1; STACK(TOP,0)←row; STACK(TOP,1)←col;
                    print row,col;
                    break;
          ]
          d←d+1;
]//end d<8
```

```
          if(d==8)
          [
              MAZE( row, col) ←'X'; pr←row; pc←col;
              TOP←TOP-1; row←STACK(TOP,0); col←STACK(TOP,1);
               MAZE( row, col) ←'O';
          ]

          print MAZE;
      ]//end TOP!=-1
      print ‘NO SOLUTION’; return FALSE;
]//end procedure
```

Working code

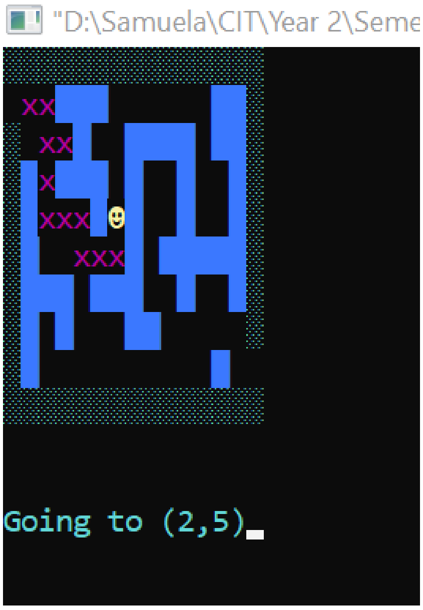
```
#include <stdio.h>
#include <unistd.h>

char maze[8][13]={  {'O',' ','#','#','#',' ',' ',' ',' ',' ',' ','#','#'},
            {' ',' ',' ','#',' ',' ','#','#','#','#',' ','#','#'},
            {'#',' ','#','#','#',' ','#',' ',' ','#',' ',' ','#'},
            {'#',' ',' ',' ','#',' ','#',' ',' ','#',' ',' ','#'},
            {'#',' ',' ',' ',' ',' ','#',' ','#','#','#','#','#'},
            {'#','#','#',' ','#','#','#',' ',' ','#',' ',' ','#'},
            {'#',' ','#',' ',' ',' ','#','#',' ',' ',' ',' ',' '},
            {'#',' ',' ',' ',' ',' ',' ',' ',' ',' ',' ','#',' '}}; //maze
int stack[104][2]={{0,0}}, top=0;
int
sol[8][13]={{1,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0},{0,0,0,0,0,0,0,0,0,0,0,0,0}}; //solution matrix
//direction order is right, down, south-east, south-west, up, north-east, north-west, left
```

```
int dir[8][2]={{0,1},{1,0},{1,1},{1,-1},{-1,0},{-1,1},{-1,-1},{0,-1}}; //possible directions

int row=0, col=0, pr, pc, end=0; //current and previous coordinates, and program end
indicator


void red(){printf("\033[1;31m");} void green(){printf("\033[1;32m");} void
yellow(){printf("\033[1;33m");} void blue(){printf("\033[1;34m");} void
purple(){printf("\033[1;35m");} void cyan(){printf("\033[1;36m");}

void screen_clear(void){system("cls");}//function to clear screen


void printMaze()//print maze

{

   int i, j;

   cyan();

printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c\n",176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176);

   for(i=0;i<8;i++)

   {

      cyan();

      if(i) printf("%c",176); else printf(" "); //making entrance and left border walls

      for(j=0;j<13;j++)

      {

         if(maze[i][j]=='O') yellow(); else if(maze[i][j]=='x') purple(); else if(maze[i][j]=='X')
red(); else blue(); //setting color

         if(maze[i][j]=='#') printf("%c",219); else if(maze[i][j]=='O') printf("%c",2); else
printf("%c",maze[i][j]);

      }

      cyan();

      if(i!=7) printf("%c\n",176); else printf(" \n"); //making exit and right border walls

   }

printf("%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c\n",176,176,176,176,176,176,176,176,
176,176,176,176,176,176,176);
```

```
}


int valid(int r, int c) //for checking if move is valid(r,c are next coordinates; row,col are current; pr, pc are previous coordinates)

{

   if(r>=0 && r<8 && c>=0 && c<13 && maze[r][c]!='#' && maze[r][c]!='x' && maze[r][c]!='X' && !(pr==r && pc==c)) return 1;

   return 0;

}


void move()

{

   int i,j,k, d=0;

   if(maze[7][12]=='O') //rat reached exit

   {

      green(); printf("\n\nSolution found\n\nSolution path:\n"); yellow();

      for(i=0; i<=top; i++)

      {

         j=stack[i][0]; k=stack[i][1]; sol[j][k]=1;

         printf("(%d,%d)\n",stack[i][0],stack[i][1]);

      }

      for(i=0;i<8;i++)

      {

         printf("\n");

         for(j=0;j<13;j++)

         {

            if(sol[i][j]==0) yellow(); else green();

            printf("%d ",sol[i][j]);

         }

      }
```

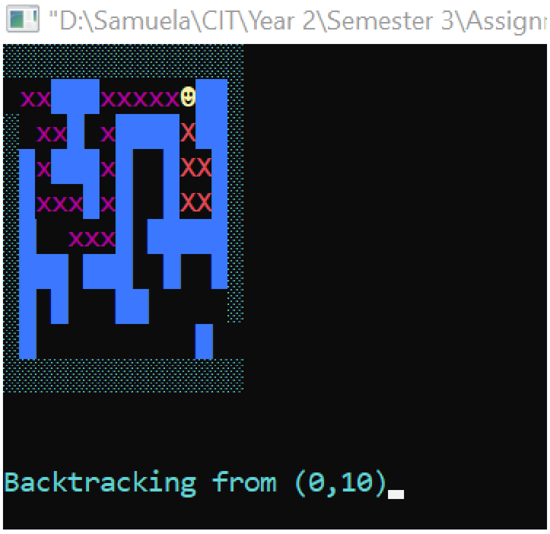
```
        end=1; return;
    }
    //moving conditions
    if(top==-1){ end=1; return; }//if no move possible then end
    while(d<8)
    {
        if(valid(row+dir[d][0],col+dir[d][1]))
        {
            maze[row][col]='x'; pr=row; pc=col;
            row=row+dir[d][0]; col=col+dir[d][1]; maze[row][col]='O';
            top++; stack[top][0]=row; stack[top][1]=col;
            printf("\n\nGoing to (%d,%d)",row,col);
            break;
        }
        d++;
    }
    if(d==8) //no free cell so backtrack
    {
        maze[row][col]='X'; pr=row; pc=col;
        printf("\n\nBacktracking from (%d,%d)",pr,pc);
        top--; row=stack[top][0]; col=stack[top][1]; maze[row][col]='O';
    }
    sleep(1); //to slow down rapid screen clearing I'm giving it a delay of 1 second
    screen_clear(); printMaze();
}

int main()//main function
{
    screen_clear(); //if I remove it then colors won't work
    printMaze(); //initial maze
    if(maze[7][12]=='#') //blocked exit
    {
        red(); printf("\nNo solution");  return -1;
    }
    while(end!=1) move(); //start moving
    if(maze[7][12]!='O'){ red(); printf("\nNo solution"); } //rat didn't reach exit

    return 0;
}
```

Output



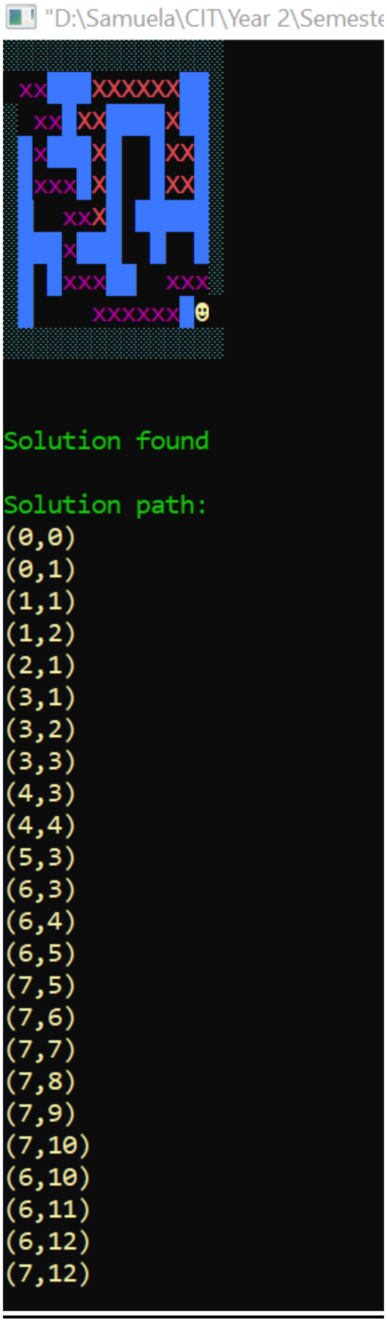
"D:\Samuela\CIT\Year 2\Seme

```
Going to (2,5)
```

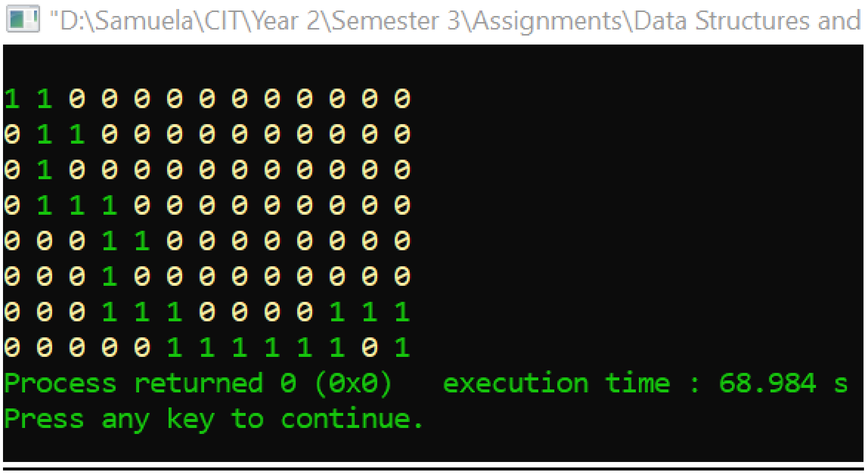


"D:\Samuela\CIT\Year 2\Semester 3\Assign

```
Backtracking from (0,10)
```



```
Solution found

Solution path:
(0,0)
(0,1)
(1,1)
(1,2)
(2,1)
(3,1)
(3,2)
(3,3)
(4,3)
(4,4)
(5,3)
(6,3)
(6,4)
(6,5)
(7,5)
(7,6)
(7,7)
(7,8)
(7,9)
(7,10)
(6,10)
(6,11)
(6,12)
(7,12)
```

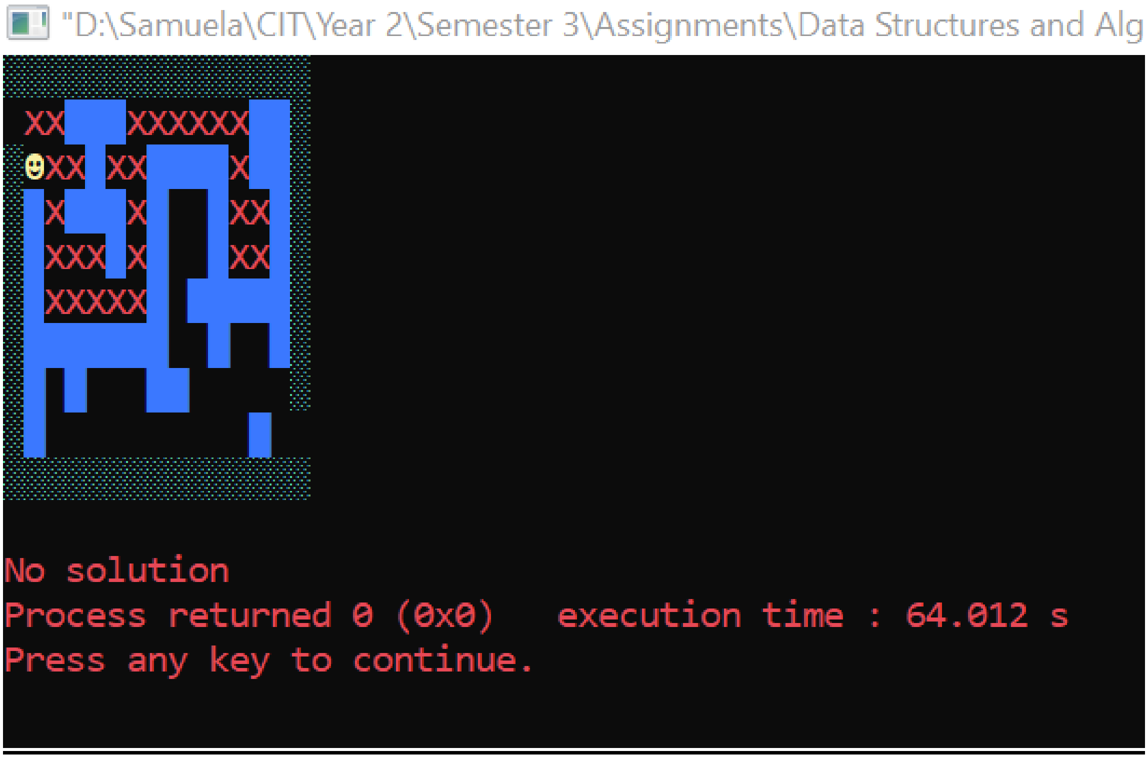


"D:\Samuela\CIT\Year 2\Semester 3\Assignments\Data Structures and

```
1 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0 1 1 1
0 0 0 0 0 1 1 1 1 1 1 0 1
Process returned 0 (0x0)   execution time : 68.984 s
Press any key to continue.
```

Samuela Abigail Mathew
71762108039
B.Tech AI&DS 3rd semester

No solution case-



"D:\Samuela\CIT\Year 2\Semester 3\Assignments\Data Structures and Alg

```
No solution
Process returned 0 (0x0)   execution time : 64.012 s
Press any key to continue.
```