CLEANING ROBOT (BACKEND ENGINE)

Subject : discrete structures

Project report

BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

3rd Semester

Submitted by :

Dhiren kinha(2k19/IT/044)

Garvit gulati(2k19/IT/049)

DEPARTMENT OF INFORMATION TECHNOLOGY DELHI TECHNOLOGICAL UNIVERSITY (Formerly Delhi College of Engineering) Bawana Road, Delhi-110042

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering) Bawana
Road, Delhi-110042

CANDIDATE’S DECLARATION

We Dhiren kinha (2k19/IT/044) and Garvit Gulati (2k19/IT/049) , students of B.tech (INFORMATION TECHNOLOGY) , hereby declare that the project Dissertation titled “cleaning robot (backend engine) ” which is submitted by us to the Department of Information Technology , Delhi Technological University , Delhi in partial fulfilment of the requirement for the award of the 3rd semester of the bachelor of technology is made by us . This work has not previously formed of the basis for the award of any degree , diploma Associateship , fellowship or other similar title or recognition .

place: Delhi
Date : 01 - 12 2020

Dhiren kinha , Garvit Gulati

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project: cleaning robot (backend engine) which is submitted by Dhiren Kinha , Roll No – 2K19/IT/044 and Garvit Gulati , Roll No – 2K19/IT/049, INFORMATION TECHNOLOGY, Delhi Technological University, Delhi in fulfillment of the requirement for the 3rd semester of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place : Delhi

Date : 01 - 12 - 2020.

supervisor :

Mrs. Swati Sharda

DEPARTMENT OF INFORMATION TECHNOLOGY

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

ACKNOWLEDGEMENT

We would like to convey our heartfelt thanks to our supervisor Mrs. Swati Sharda for her ingenious ideas, tremendous help and cooperation. We are extremely grateful to our friends who gave valuable suggestions and guidance for completion of our project. The cooperation and healthy criticism came handy and useful with them.

INTRODUCTION :

In this project, we aim to implement the backend engine for a house cleaning robot. The robot will scan the map of a room/floor which is assigned to it. Then it will take the destination to be cleaned along with the amount of dust on it and then generate an optimum path cleaning all the locations preferring them according to the dust reach there by dodging the obstacles of the room efficiently and will clean it.

We have used the concept of graphs and A* star search algorithm to find the path. Even though we are using platform specific features like coloured output and clear screen, we have made the code platform independent with the help of C++ pre-processor directives.

Overview:

In this project, we are implementing a household helper robot’s backend system. The robot will scan the map of a room/floor which is assigned to it.Then it will take the destination to be cleaned along with the amount of dust on it and then generate an optimum path cleaning all the locations preferring them according to the dust reach there by dodging the obstacles of the room efficiently and will clean it.

Functionalities:

We will incorporate the following functionalities in our code:

1.Along with the location to be cleaned we will also consider the amount of dust or the cleanliness required at the location and accordingly the locations will be arranged in a preference order and then the distance would be found between them and according a path will be displayed incorporating all the locations.

2.We will provide an option to configure the home location which will be considered as the staying location for the robot so that after cleaning all the locations it is not in the way and resides safely in a corner without affecting anyone.

3. Finally the user will also be able to edit the map easily via a .txt file without requiring any knowledge about coding or anything. This is crucial as it is often that we change the setting of our room or the floor map and the robot will be useless so to overcome this user can just open the floor map file and

define the new obstacles and free space with a combination of 1s and 0s without any problem.

4. We will also display the path taken by the robot in the grid in the form of 0s and 1s.

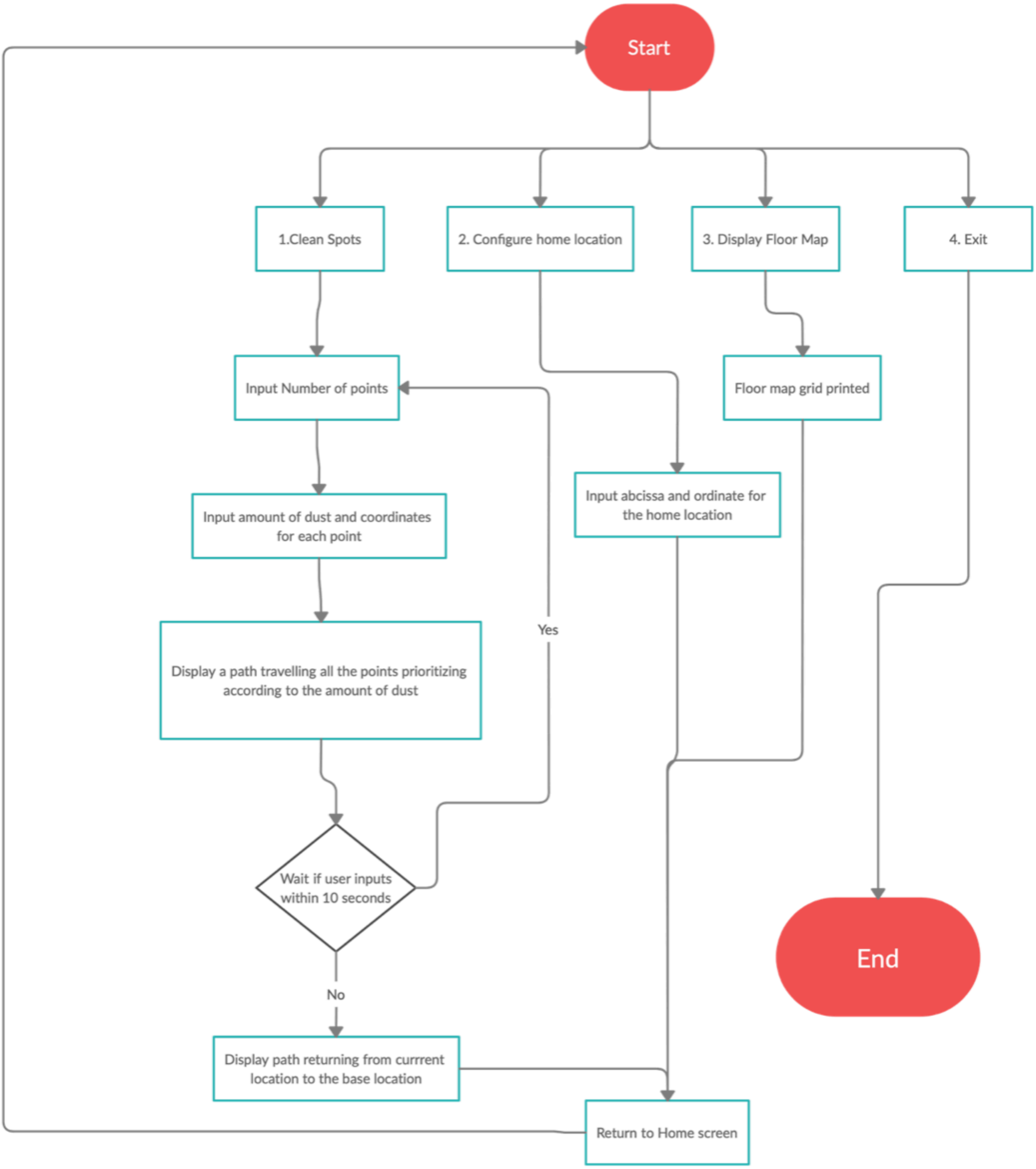
Features :

Colors: To make the input and output more interactive, we would be adding colors and styles like bold and italics to the text. This will help in making the code more user friendly.

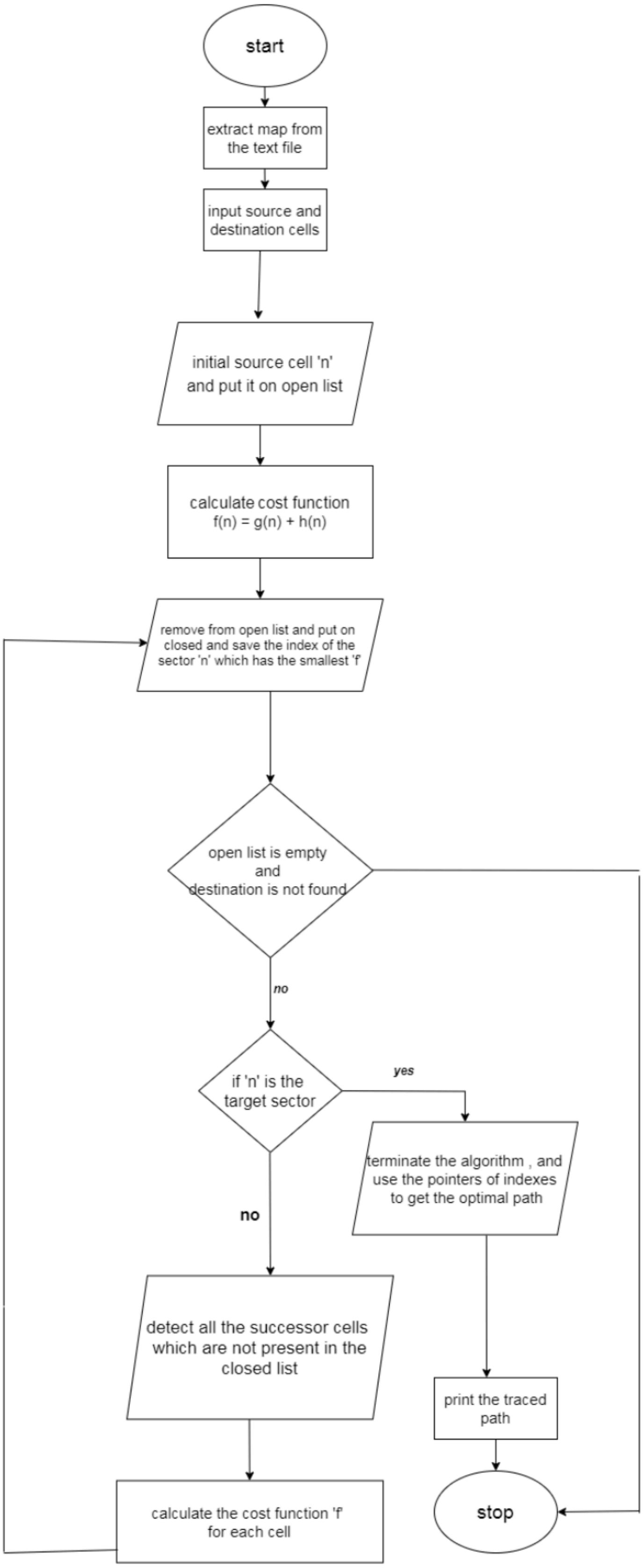
Screen Changing: To make the output more readable we have implemented our own clear screen function which will clear the screen after input and output.

Cross-Platform: The above mentioned features have different implementation for windows , mac and linux and therefore, restricting the code to only one platform but with the help of c++ preprocessor directives we would recognise the system on which our code is running and would accordingly use the implementation of the feature so as to improve the portability of the code.

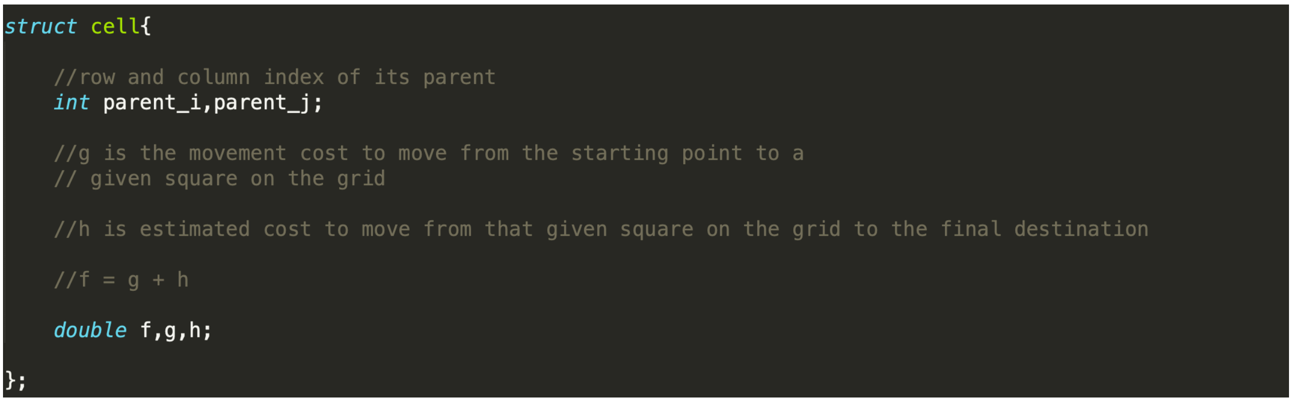
FLOWCHART :


Start
1.Clean Spots
2. Configure home location
3. Display Floor Map
4. Exit
Input Number of points
Floor map grid printed
Input amount of dust and coordinates for each point
Input abcissa and ordinate for the home location
Yes
Display a path travelling all the points prioritizing according to the amount of dust
Wait if user inputs within 10 seconds
End
No
Display path returning from currrent location to the base location
Return to Home screen

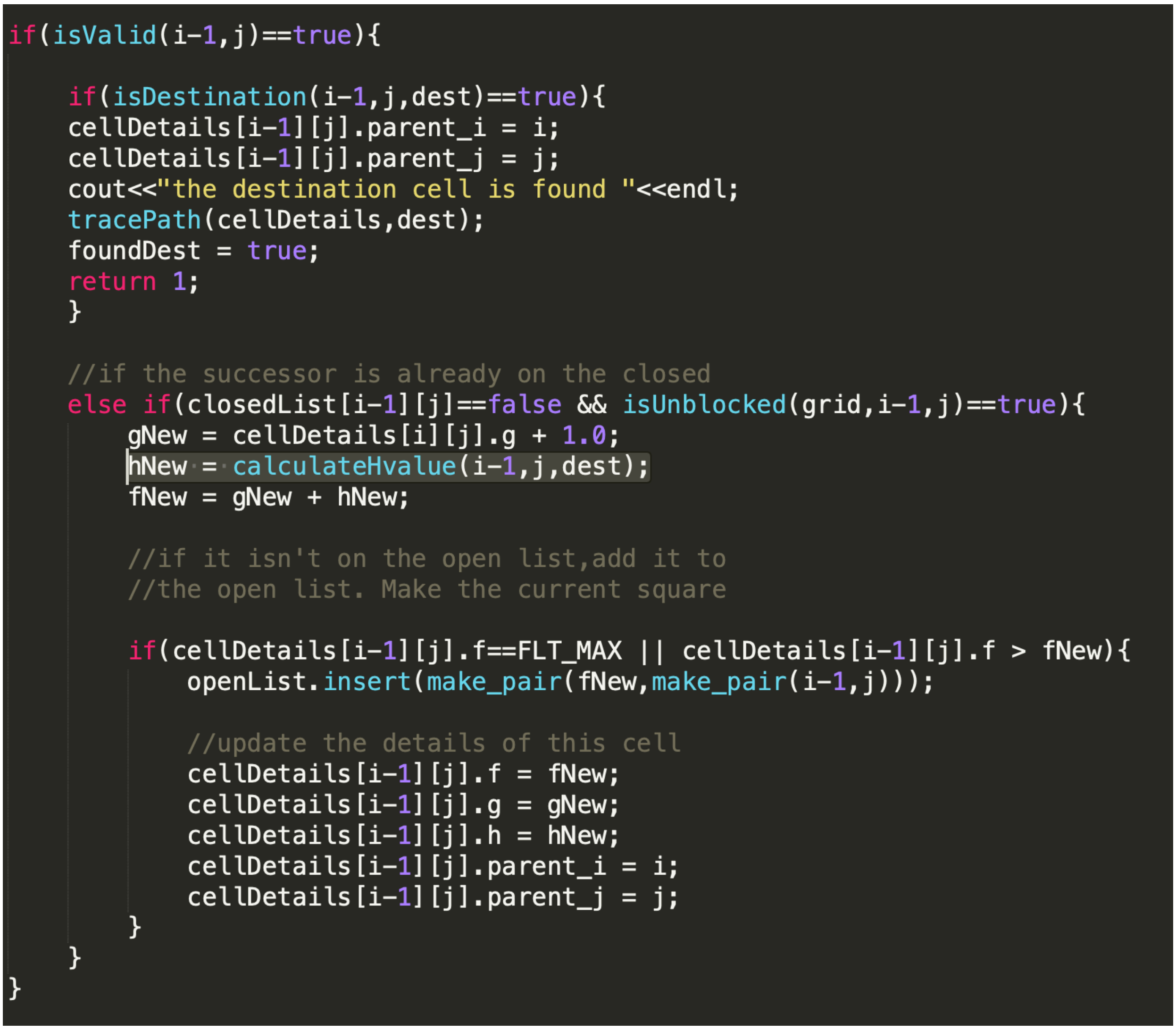
FLOWCHART FOR A* ALGORITHM:


start
extract map from the text file
input source and destination cells
initial source cell 'n' and put it on open list
calculate cost function f(n) = g(n) + h(n)
remove from open list and put on closed and save the index of the sector 'n' which has the smallest 'f'
open list is empty and destination is not found
no
if 'n' is the target sector
yes
terminate the algorithm , and use the pointers of indexes to get the optimal path
no
detect all the successor cells which are not present in the closed list
print the traced path
calculate the cost function 'f' for each cell
stop

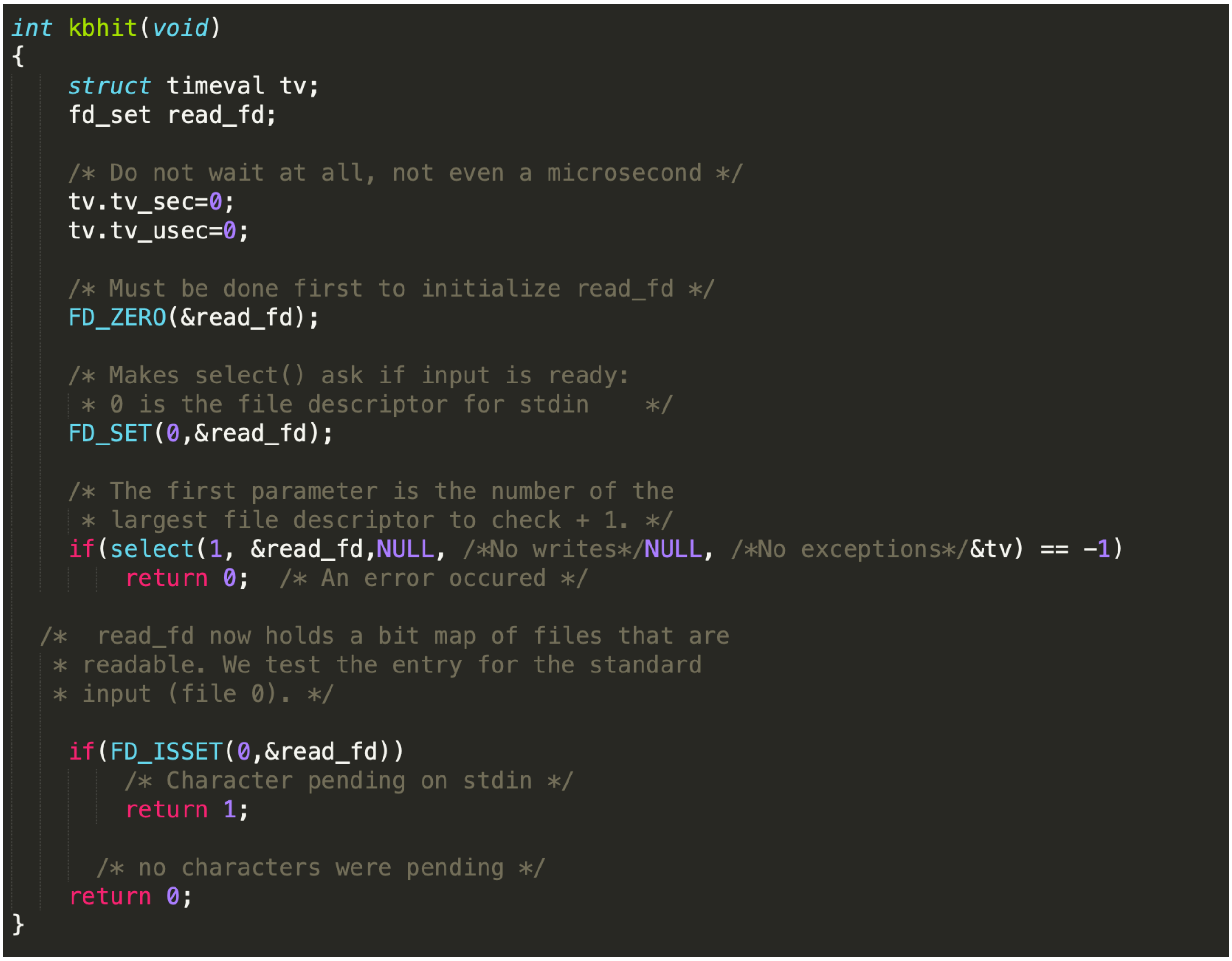
CODE SNIPPETS:



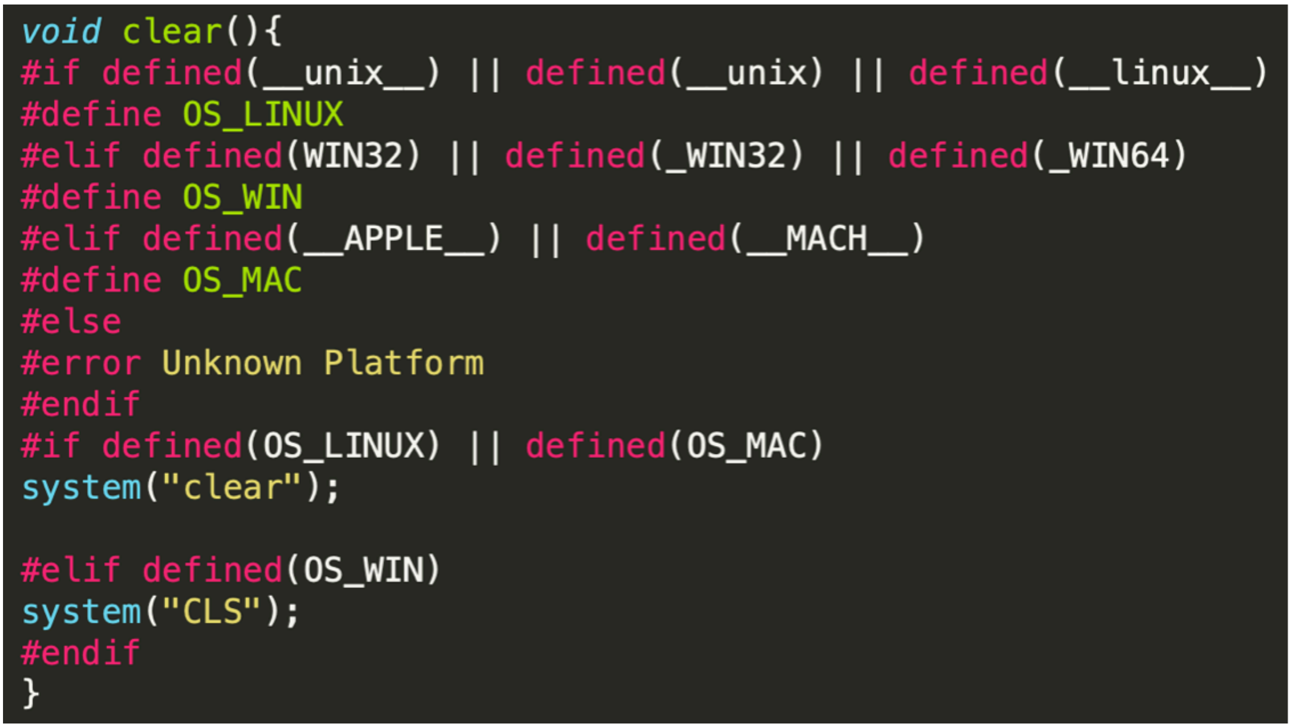
```
struct cell{

    //row and column index of its parent
    int parent_i,parent_j;

    //g is the movement cost to move from the starting point to a
    // given square on the grid

    //h is estimated cost to move from that given square on the grid to the final destination

    //f = g + h

    double f,g,h;

};
```



```
if(isValid(i-1,j)==true){

    if(isDestination(i-1,j,dest)==true){
    cellDetails[i-1][j].parent_i = i;
    cellDetails[i-1][j].parent_j = j;
    cout<<"the destination cell is found "<<endl;
    tracePath(cellDetails,dest);
    foundDest = true;
    return 1;
    }

    //if the successor is already on the closed
    else if(closedList[i-1][j]==false && isUnblocked(grid,i-1,j)==true){
        gNew = cellDetails[i][j].g + 1.0;
        hNew = calculateHvalue(i-1,j,dest);
        fNew = gNew + hNew;

        //if it isn't on the open list,add it to
        //the open list. Make the current square

        if(cellDetails[i-1][j].f==FLT_MAX || cellDetails[i-1][j].f > fNew){
            openList.insert(make_pair(fNew,make_pair(i-1,j)));

            //update the details of this cell
            cellDetails[i-1][j].f = fNew;
            cellDetails[i-1][j].g = gNew;
            cellDetails[i-1][j].h = hNew;
            cellDetails[i-1][j].parent_i = i;
            cellDetails[i-1][j].parent_j = j;
        }
    }
}
```

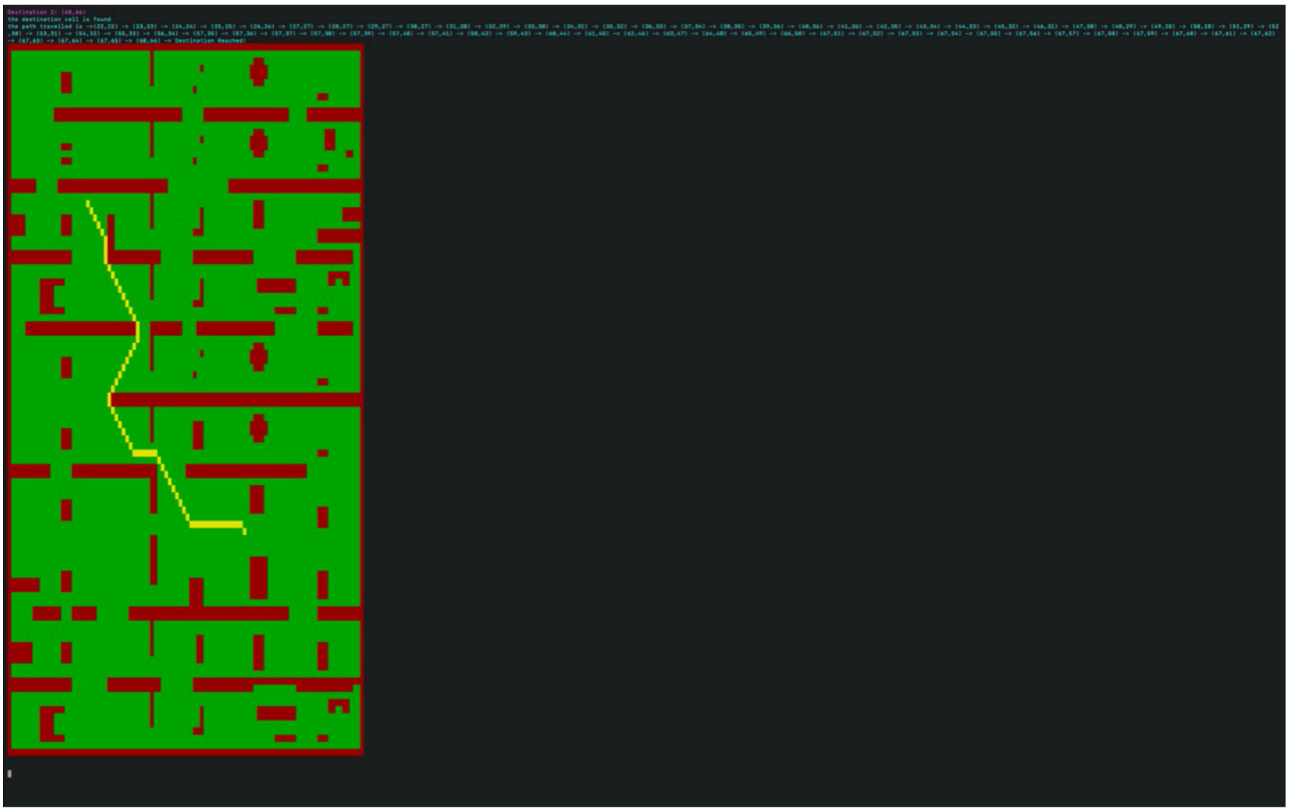


```
int kbhit(void)
{
    struct timeval tv;
    fd_set read_fd;

    /* Do not wait at all, not even a microsecond */
    tv.tv_sec=0;
    tv.tv_usec=0;

    /* Must be done first to initialize read_fd */
    FD_ZERO(&read_fd);

    /* Makes select() ask if input is ready:
     * 0 is the file descriptor for stdin    */
    FD_SET(0,&read_fd);

    /* The first parameter is the number of the
     * largest file descriptor to check + 1. */
    if(select(1, &read_fd,NULL, /*No writes*/NULL, /*No exceptions*/&tv) == -1)
        return 0;  /* An error occured */

  /*  read_fd now holds a bit map of files that are
   * readable. We test the entry for the standard
   * input (file 0). */

    if(FD_ISSET(0,&read_fd))
        /* Character pending on stdin */
        return 1;

      /* no characters were pending */
    return 0;
}
```

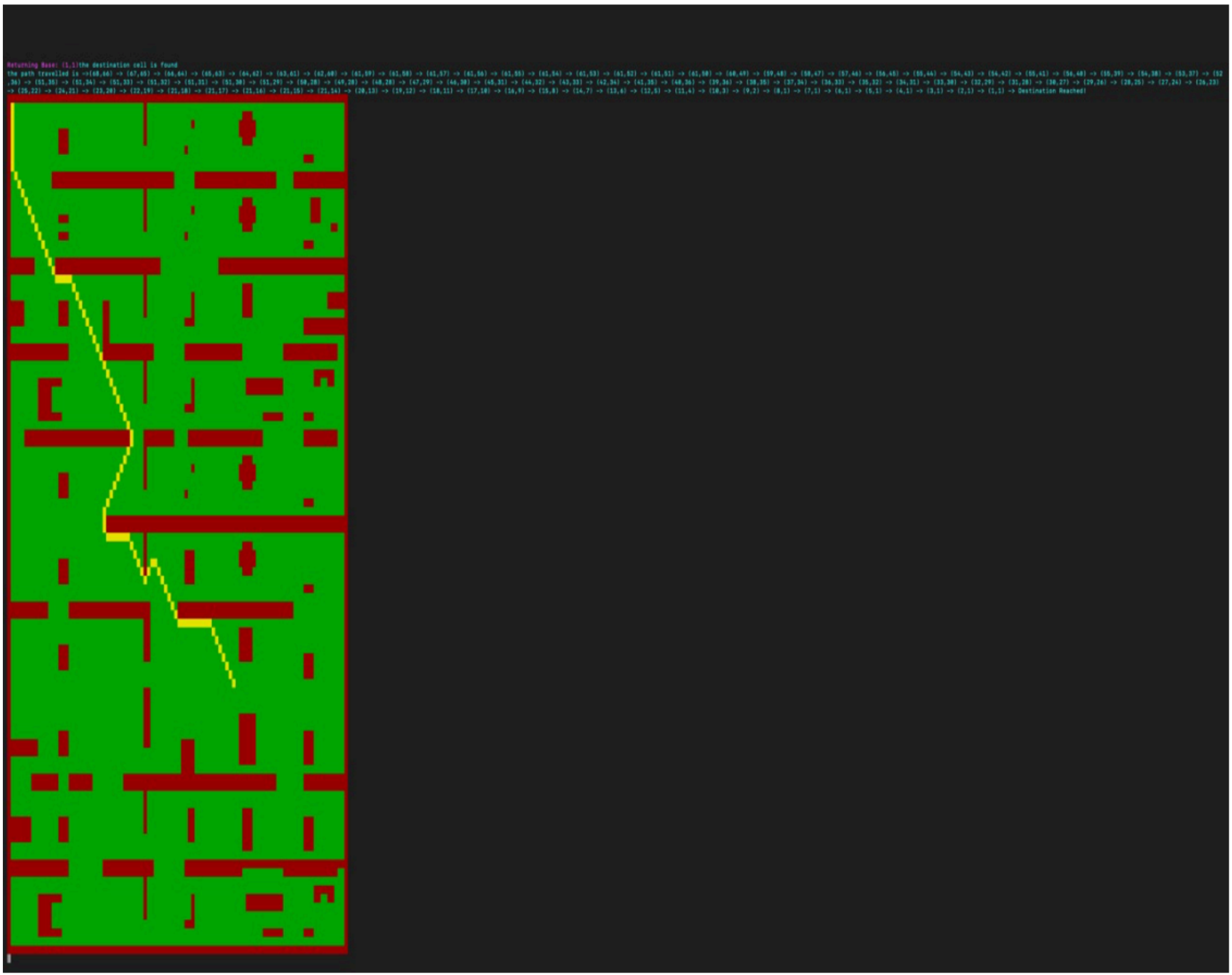


```
void clear(){
#if defined(__unix__) || defined(__unix) || defined(__linux__)
#define OS_LINUX
#elif defined(WIN32) || defined(_WIN32) || defined(_WIN64)
#define OS_WIN
#elif defined(__APPLE__) || defined(__MACH__)
#define OS_MAC
#else
#error Unknown Platform
#endif
#if defined(OS_LINUX) || defined(OS_MAC)
system("clear");

#elif defined(OS_WIN)
system("CLS");
#endif
}
```

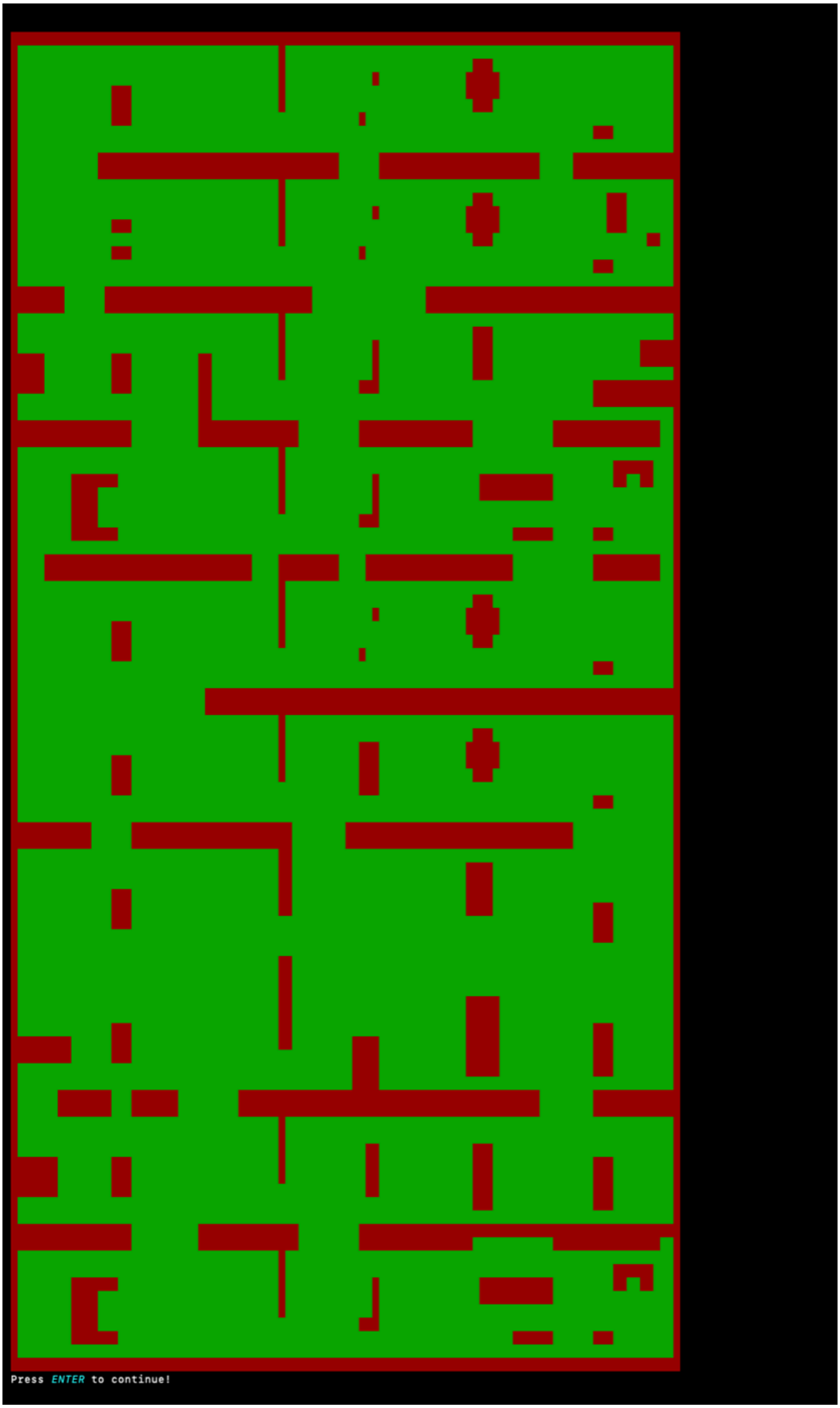
OUTPUT:



```
Destination 2: (68,66)
the destination cell is found
the path travelled is ->(22,22) -> (23,23) -> (24,24) -> (25,25) -> (26,26) -> (27,27) -> (28,27) -> (29,27) -> (30,27) -> (31,28) -> (32,29) -> (33,30) -> (34,31) -> (35,32) -> (36,33) -> (37,34) -> (38,35) -> (39,36) -> (40,36) -> (41,36) -> (42,35) -> (43,34) -> (44,33) -> (45,32) -> (46,31) -> (47,30) -> (48,29) -> (49,28) -> (50,28) -> (51,29) -> (52
,30) -> (53,31) -> (54,32) -> (55,33) -> (56,34) -> (57,35) -> (57,36) -> (57,37) -> (57,38) -> (57,39) -> (57,40) -> (57,41) -> (58,42) -> (59,43) -> (60,44) -> (61,45) -> (62,46) -> (63,47) -> (64,48) -> (65,49) -> (66,50) -> (67,51) -> (67,52) -> (67,53) -> (67,54) -> (67,55) -> (67,56) -> (67,57) -> (67,58) -> (67,59) -> (67,60) -> (67,61) -> (67,62)
-> (67,63) -> (67,64) -> (67,65) -> (68,66) -> Destination Reached!
```



```
Returning Base: (1,1)the destination cell is found
the path travelled is ->(68,66) -> (67,65) -> (66,64) -> (65,63) -> (64,62) -> (63,61) -> (62,60) -> (61,59) -> (61,58) -> (61,57) -> (61,56) -> (61,55) -> (61,54) -> (61,53) -> (61,52) -> (61,51) -> (61,50) -> (60,49) -> (59,48) -> (58,47) -> (57,46) -> (56,45) -> (55,44) -> (54,43) -> (54,42) -> (55,41) -> (56,40) -> (55,39) -> (54,38) -> (53,37) -> (52
,36) -> (51,35) -> (51,34) -> (51,33) -> (51,32) -> (51,31) -> (51,30) -> (51,29) -> (50,28) -> (49,28) -> (48,28) -> (47,29) -> (46,30) -> (45,31) -> (44,32) -> (43,33) -> (42,34) -> (41,35) -> (40,36) -> (39,36) -> (38,35) -> (37,34) -> (36,33) -> (35,32) -> (34,31) -> (33,30) -> (32,29) -> (31,28) -> (30,27) -> (29,26) -> (28,25) -> (27,24) -> (26,23)
-> (25,22) -> (24,21) -> (23,20) -> (22,19) -> (21,18) -> (21,17) -> (21,16) -> (21,15) -> (21,14) -> (20,13) -> (19,12) -> (18,11) -> (17,10) -> (16,9) -> (15,8) -> (14,7) -> (13,6) -> (12,5) -> (11,4) -> (10,3) -> (9,2) -> (8,1) -> (7,1) -> (6,1) -> (5,1) -> (4,1) -> (3,1) -> (2,1) -> (1,1) -> Destination Reached!
```



```
Press ENTER to continue!
```