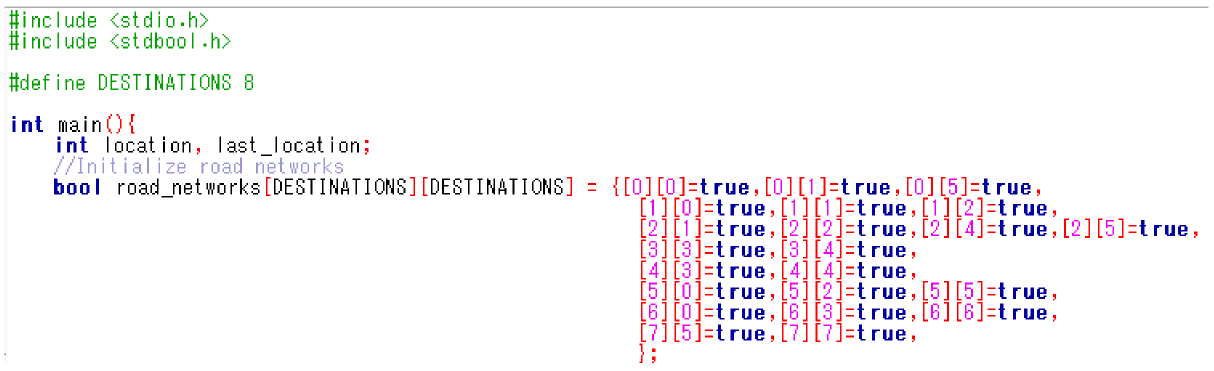
Name: Carl Jorenz Gimeno   Section:1

Solution:

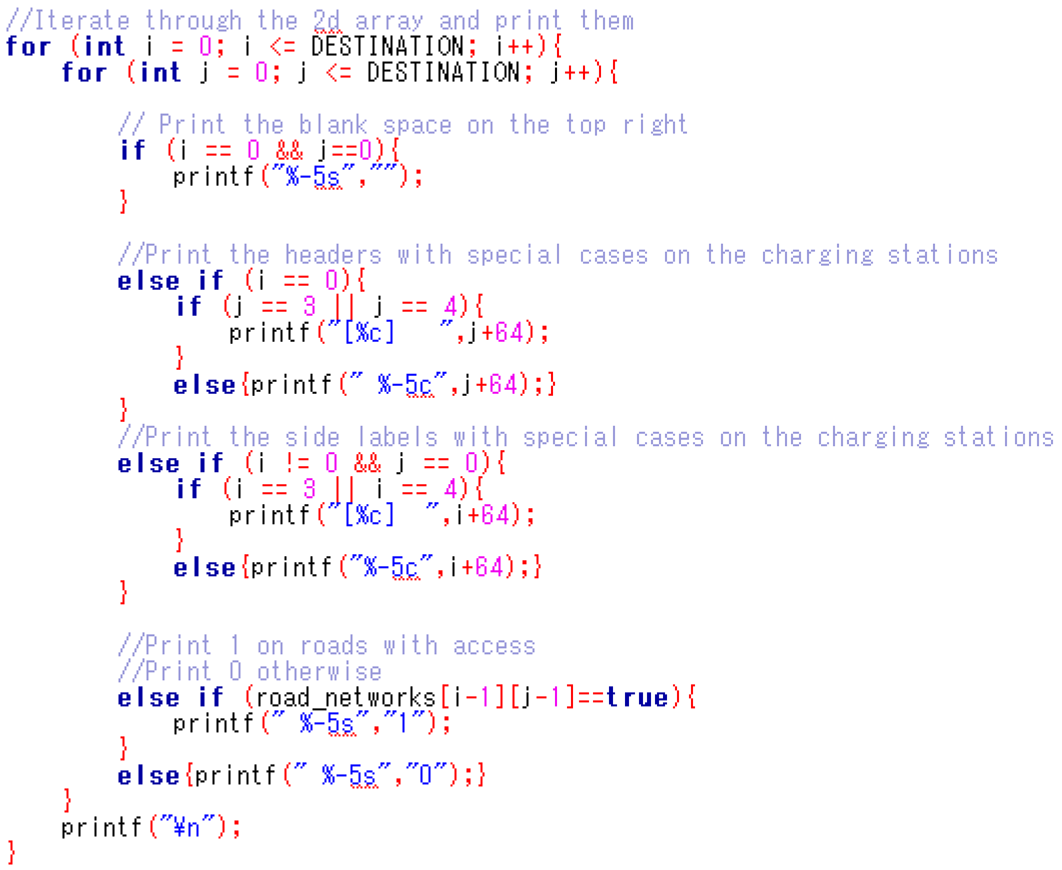
I. Initialization



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

#define DESTINATIONS 8

int main(){
    int location, last_location;
    //Initialize road networks
    bool road_networks[DESTINATIONS][DESTINATIONS] = {[0][0]=true,[0][1]=true,[0][5]=true,
                                                      [1][0]=true,[1][1]=true,[1][2]=true,
                                                      [2][1]=true,[2][2]=true,[2][4]=true,[2][5]=true,
                                                      [3][3]=true,[3][4]=true,
                                                      [4][3]=true,[4][4]=true,
                                                      [5][0]=true,[5][2]=true,[5][5]=true,
                                                      [6][0]=true,[6][3]=true,[6][6]=true,
                                                      [7][5]=true,[7][7]=true,
                                                      };
```

The 2d array and two variables are initialized. The 2d array will hold the values of the pathways. The variable location holds the value of the user's current position relative to the pathways while the variable last_location holds the value of the user's last position and will serve to prevent backtracking and softlock the user in an endless loop.

II. 2D Array Display



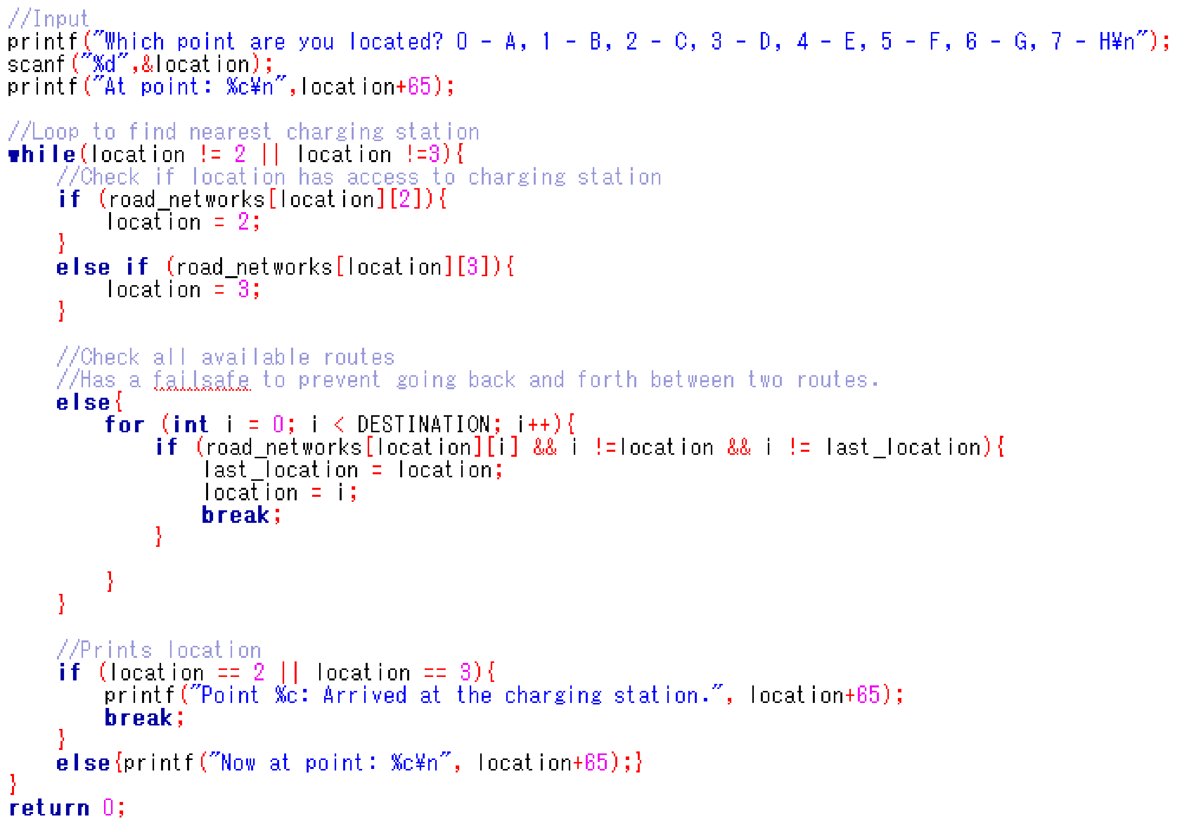
```c
//Iterate through the 2d array and print them
for (int i = 0; i <= DESTINATION; i++){
    for (int j = 0; j <= DESTINATION; j++){

        // Print the blank space on the top right
        if (i == 0 && j==0){
            printf("%-5s","");
        }

        //Print the headers with special cases on the charging stations
        else if (i == 0){
            if (j == 3 || j == 4){
                printf("[%c]   ",j+64);
            }
            else{printf(" %-5c",j+64);}
        }
        //Print the side labels with special cases on the charging stations
        else if (i != 0 && j == 0){
            if (i == 3 || i == 4){
                printf("[%c]  ",i+64);
            }
            else{printf("%-5c",i+64);}
        }

        //Print 1 on roads with access
        //Print 0 otherwise
        else if (road_networks[i-1][j-1]==true){
            printf(" %-5s","1");
        }
        else{printf(" %-5s","0");}
    }
    printf("¥n");
}
```

During the printing of the table, the first row and column are reserved for the headings. The headers are made by adding 64 to the index and outputting them as characters.

Name: Carl Jorenz Gimeno Section:1

On the headers for the charging stations, C and D, brackets were added. At first, I'd opted for static headers, but the necessary code is very long, and this set-up allows for dynamic array sizes.

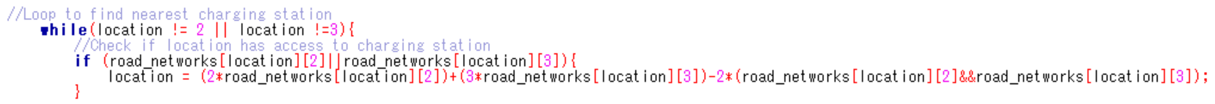
III. I/O



```
//Input
printf("Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H¥n");
scanf("%d",&location);
printf("At point: %c¥n",location+65);

//Loop to find nearest charging station
while(location != 2 || location !=3){
    //Check if location has access to charging station
    if (road_networks[location][2]){
        location = 2;
    }
    else if (road_networks[location][3]){
        location = 3;
    }

    //Check all available routes
    //Has a failsafe to prevent going back and forth between two routes.
    else{
        for (int i = 0; i < DESTINATION; i++){
            if (road_networks[location][i] && i !=location && i != last_location){
                last_location = location;
                location = i;
                break;
            }

        }
    }

    //Prints location
    if (location == 2 || location == 3){
        printf("Point %c: Arrived at the charging station.", location+65);
        break;
    }
    else{printf("Now at point: %c¥n", location+65);}
}
return 0;
```

Scans for input and saves them to the variable location. Print out the location of the user by adding 65 to location and converting it into char.

For the main loop, it will first check if there's an open road to a charging station. At first, I would like to use this algorithm:



```
//Loop to find nearest charging station
    while(location != 2 || location !=3){
        //Check if location has access to charging station
        if (road_networks[location][2]||road_networks[location][3]){
            location = (2*road_networks[location][2])+(3*road_networks[location][3])-2*(road_networks[location][2]&&road_networks[location][3]);
        }
```

It would reduce the code into one line, but it is very hard to parse. All it does is get the value of the road, whether it's open or not, and multiply it with their position and subtract 2 if both roads are open.

The next part checks for all available routes sequentially until it finds an open road that it didn't travel to previously.

The last part prints the current location and a special text when on a charging station.