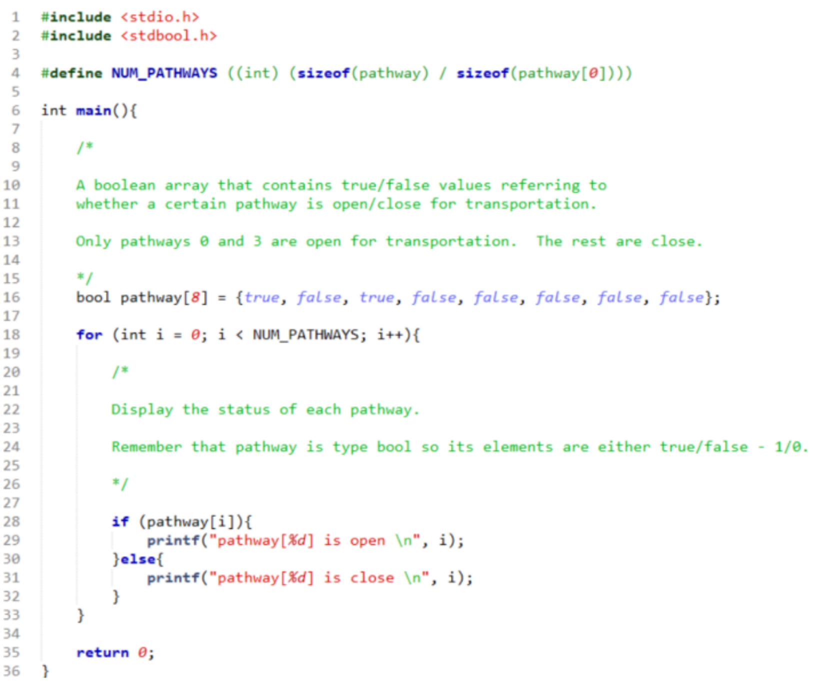
Neil Bryan F. Flauta CMSC 21 -1

Lectures 6-7 Assignment

1. In the program below, an array named pathway contains eight bool values. Each bool element refers to whether a pathway is open or close for transportation.

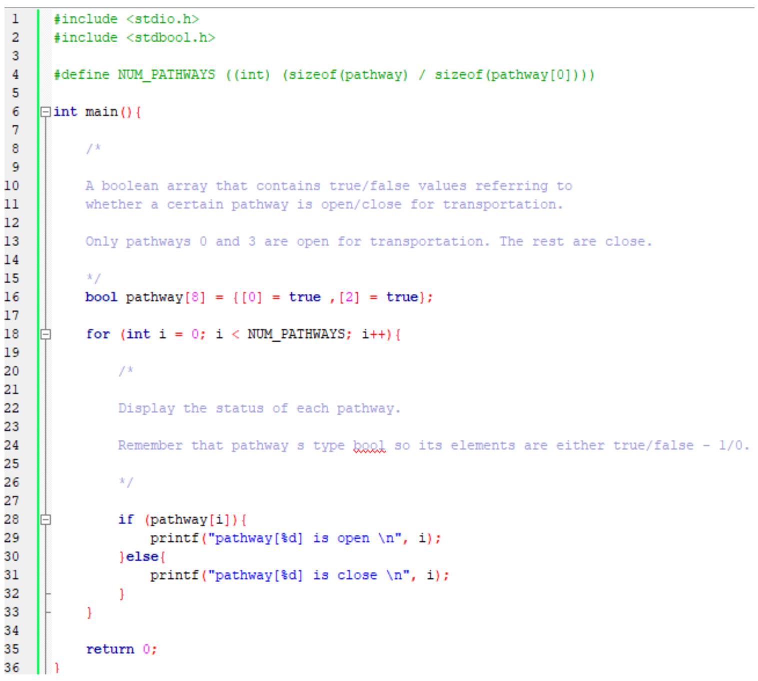
Only pathways 0 and 2 are open while the rest are still close due to road constructions and fixings.



```c
1   #include <stdio.h>
2   #include <stdbool.h>
3
4   #define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))
5
6   int main(){
7
8       /*
9
10      A boolean array that contains true/false values referring to
11      whether a certain pathway is open/close for transportation.
12
13      Only pathways 0 and 3 are open for transportation.  The rest are close.
14
15      */
16      bool pathway[8] = {true, false, true, false, false, false, false, false};
17
18      for (int i = 0; i < NUM_PATHWAYS; i++){
19
20          /*
21
22          Display the status of each pathway.
23
24          Remember that pathway is type bool so its elements are either true/false - 1/0.
25
26          */
27
28          if (pathway[i]){
29              printf("pathway[%d] is open \n", i);
30          }else{
31              printf("pathway[%d] is close \n", i);
32          }
33      }
34
35      return 0;
36  }
```

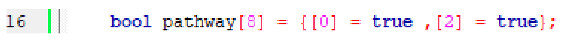
a. Revise line 16 such that you use a designated initializer to set pathways 0 and 2 to true, and the rest will be false. Make the initializer as short as possible.

Screenshot of the Code:



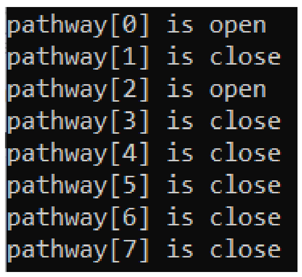
```c
1   #include <stdio.h>
2   #include <stdbool.h>
3
4   #define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))
5
6   int main(){
7
8       /*
9
10      A boolean array that contains true/false values referring to
11      whether a certain pathway is open/close for transportation.
12
13      Only pathways 0 and 3 are open for transportation. The rest are close.
14
15      */
16      bool pathway[8] = {[0] = true ,[2] = true};
17
18      for (int i = 0; i < NUM_PATHWAYS; i++){
19
20          /*
21
22          Display the status of each pathway.
23
24          Remember that pathway s type bool so its elements are either true/false - 1/0.
25
26          */
27
28          if (pathway[i]){
29              printf("pathway[%d] is open \n", i);
30          }else{
31              printf("pathway[%d] is close \n", i);
32          }
33      }
34
35      return 0;
36  }
```

Line 16:



```c
16      bool pathway[8] = {[0] = true ,[2] = true};
```

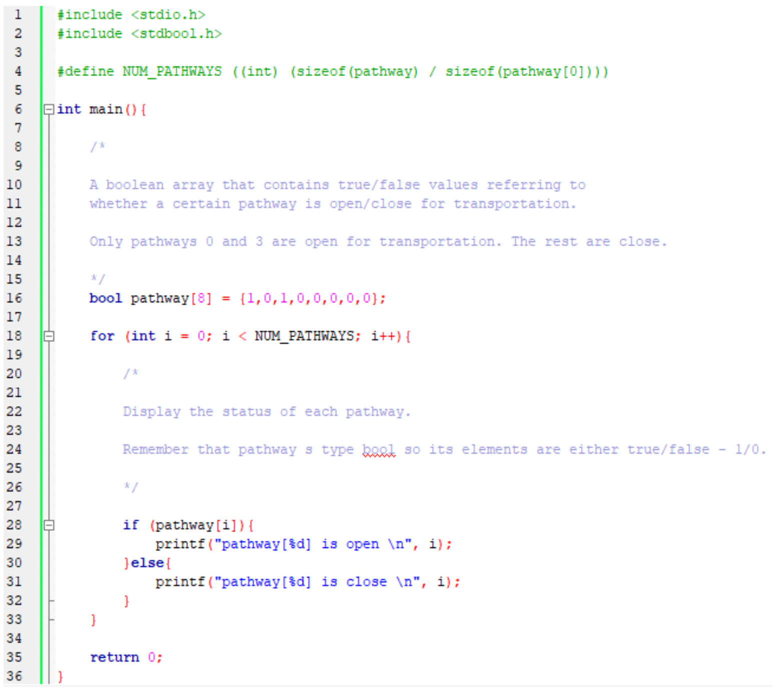
Output:



```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

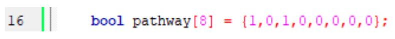
b. Revise line 16 such that the initializer will be short as possible (without using a designated initializer).

Screenshot of the code:



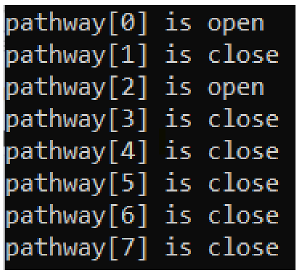
```
1   #include <stdio.h>
2   #include <stdbool.h>
3
4   #define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))
5
6   int main(){
7
8       /*
9
10      A boolean array that contains true/false values referring to
11      whether a certain pathway is open/close for transportation.
12
13      Only pathways 0 and 3 are open for transportation. The rest are close.
14
15      */
16      bool pathway[8] = {1,0,1,0,0,0,0,0};
17
18      for (int i = 0; i < NUM_PATHWAYS; i++){
19
20          /*
21
22          Display the status of each pathway.
23
24          Remember that pathway s type bool so its elements are either true/false - 1/0.
25
26          */
27
28          if (pathway[i]){
29              printf("pathway[%d] is open \n", i);
30          }else{
31              printf("pathway[%d] is close \n", i);
32          }
33      }
34
35      return 0;
36  }
```

Line 16:



```
16      bool pathway[8] = {1,0,1,0,0,0,0,0};
```

Output:



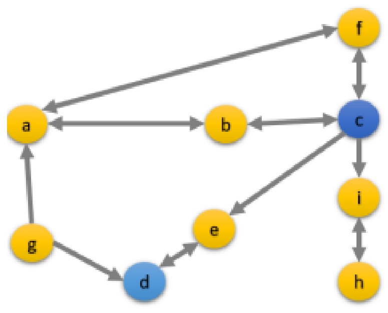
```
pathway[0] is open
pathway[1] is close
pathway[2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

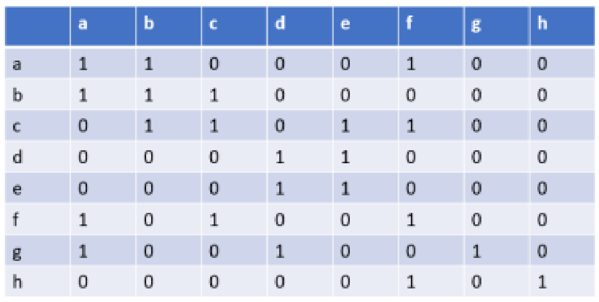
2. A road network can be represented using graphs. Assuming we have points / stations a, b, c, d, e, f, g, and h, we can represent a direct path from a point to another point using arrows.
For example, based on the graph below:

- There is a two-way path between point a and point b, point a and point f, point f and point c, and point d and e

- There is a one-way path from point c to point i but no direct path between point i to point c.

All of the nodes are points/destinations, but the yellow ones specifically represent charging stations. The road network between these points/destinations can be represented using an adjacency matrix of Booleans (0s and 1s), as shown below. For instance, a→ b = 1 and b → a = 1 given that there's a two-way direct path between a and b. Meanwhile, a → c = 0 since there is no direct path between a and c. Moreover, a → g = 0 but g → a = 1 since there is a one-way path from point g to point a.



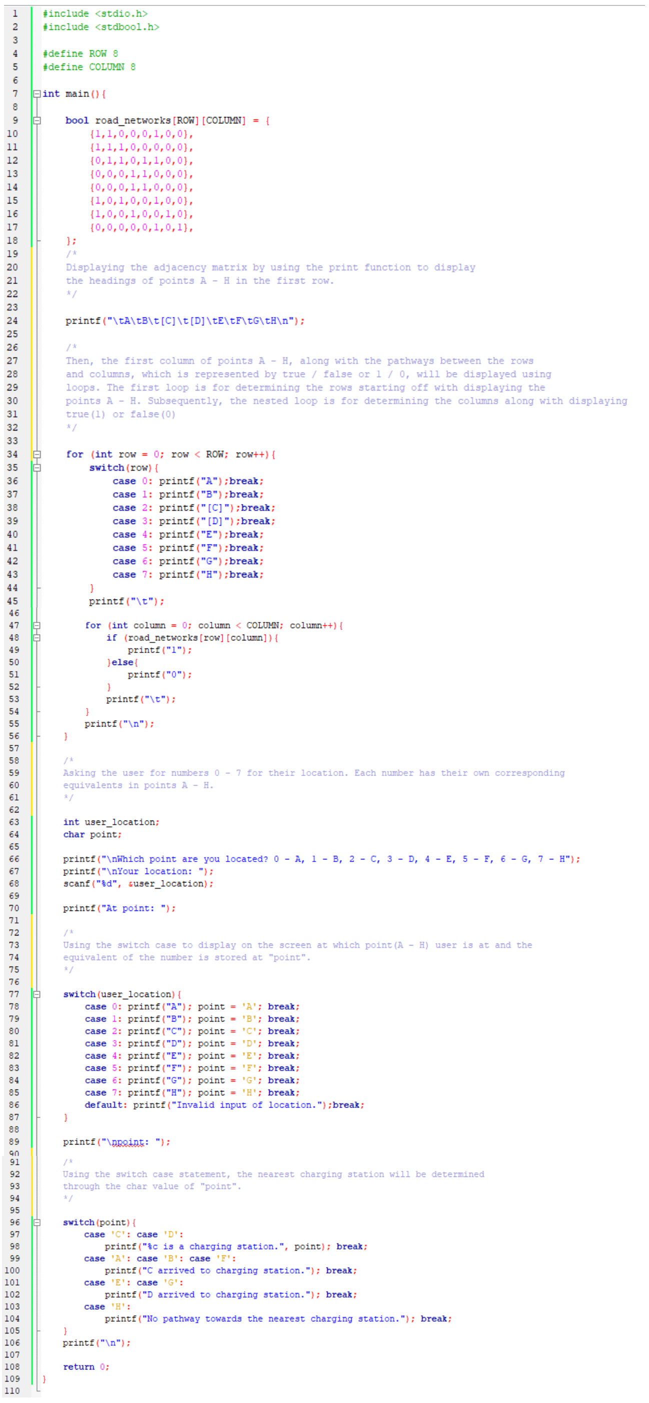


	a	b	c	d	e	f	g	h
a	1	1	0	0	0	1	0	0
b	1	1	1	0	0	0	0	0
c	0	1	1	0	1	1	0	0
d	0	0	0	1	1	0	0	0
e	0	0	0	1	1	0	0	0
f	1	0	1	0	0	1	0	0
g	1	0	0	1	0	0	1	0
h	0	0	0	0	0	1	0	1

As a programming assignment:

1. Declare and initialize a road_networks multidimensional array that represents the adjacency matrix
2. Display the adjacency matrix. Put a bracket to the points/destinations that are considered as charging stations, e.g. [c], [d]
3. Given a point / destination, determine the nearest charging station. For example, if you are in point a, the nearest charging station is point c. If you are in point e, the nearest charging station is point d.
4. Bonus: Use a macro to define the size of the 2d array

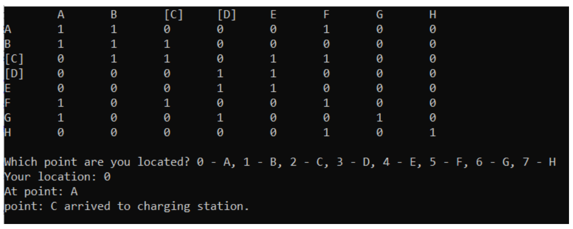
Screenshot of the code:



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

#define ROW 8
#define COLUMN 8

int main(){

    bool road_networks[ROW][COLUMN] = {
        {1,1,0,0,0,1,0,0},
        {1,1,1,0,0,0,0,0},
        {0,1,1,0,1,1,0,0},
        {0,0,0,1,1,0,0,0},
        {0,0,0,1,1,0,0,0},
        {1,0,1,0,0,1,0,0},
        {1,0,0,1,0,0,1,0},
        {0,0,0,0,0,1,0,1},
    };
    /*
    Displaying the adjacency matrix by using the print function to display
    the headings of points A - H in the first row.
    */

    printf("\tA\tB\t[C]\t[D]\tE\tF\tG\tH\n");

    /*
    Then, the first column of points A - H, along with the pathways between the rows
    and columns, which is represented by true / false or 1 / 0, will be displayed using
    loops. The first loop is for determining the rows starting off with displaying the
    points A - H. Subsequently, the nested loop is for determining the columns along with displaying
    true(1) or false(0)
    */

    for (int row = 0; row < ROW; row++){
        switch(row){
            case 0: printf("A");break;
            case 1: printf("B");break;
            case 2: printf("[C]");break;
            case 3: printf("[D]");break;
            case 4: printf("E");break;
            case 5: printf("F");break;
            case 6: printf("G");break;
            case 7: printf("H");break;
        }
        printf("\t");

       for (int column = 0; column < COLUMN; column++){
           if (road_networks[row][column]){
               printf("1");
           }else{
               printf("0");
           }
           printf("\t");
       }
       printf("\n");
    }

   /*
   Asking the user for numbers 0 - 7 for their location. Each number has their own corresponding
   equivalents in points A - H.
   */

   int user_location;
   char point;

   printf("\nWhich point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H");
   printf("\nYour location: ");
   scanf("%d", &user_location);

   printf("At point: ");

   /*
   Using the switch case to display on the screen at which point(A - H) user is at and the
   equivalent of the number is stored at "point".
   */

   switch(user_location){
       case 0: printf("A"); point = 'A'; break;
       case 1: printf("B"); point = 'B'; break;
       case 2: printf("C"); point = 'C'; break;
       case 3: printf("D"); point = 'D'; break;
       case 4: printf("E"); point = 'E'; break;
       case 5: printf("F"); point = 'F'; break;
       case 6: printf("G"); point = 'G'; break;
       case 7: printf("H"); point = 'H'; break;
       default: printf("Invalid input of location.");break;
   }

   printf("\npoint: ");

   /*
   Using the switch case statement, the nearest charging station will be determined
   through the char value of "point".
   */

   switch(point){
       case 'C': case 'D':
           printf("%c is a charging station.", point); break;
       case 'A': case 'B': case 'F':
           printf("C arrived to charging station."); break;
       case 'E': case 'G':
           printf("D arrived to charging station."); break;
       case 'H':
           printf("No pathway towards the nearest charging station."); break;
   }
   printf("\n");

   return 0;
}
```

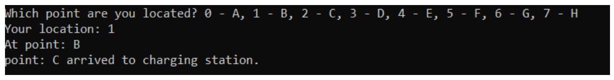
Example Outputs:

Origin at Point A along with the Matrix



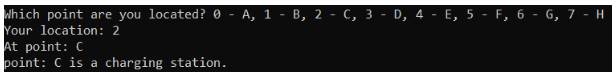
```
        A       B       [C]     [D]     E       F       G       H
A       1       1       0       0       0       1       0       0
B       1       1       1       0       0       0       0       0
[C]     0       1       1       0       1       1       0       0
[D]     0       0       0       1       1       0       0       0
E       0       0       0       1       1       0       0       0
F       1       0       1       0       0       1       0       0
G       1       0       0       1       0       0       1       0
H       0       0       0       0       0       1       0       1

Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 0
At point: A
point: C arrived to charging station.
```

Origin at Point B



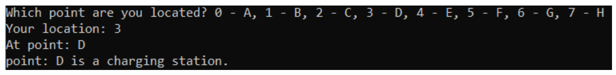
```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 1
At point: B
point: C arrived to charging station.
```

Origin at Point C



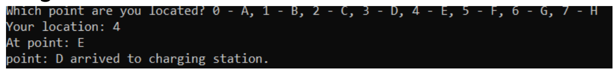
```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 2
At point: C
point: C is a charging station.
```

Origin at Point D



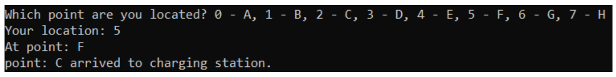
```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 3
At point: D
point: D is a charging station.
```

Origin at Point E



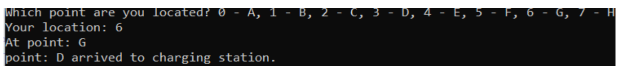
```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 4
At point: E
point: D arrived to charging station.
```

Origin at Point F



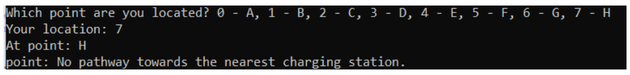
```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 5
At point: F
point: C arrived to charging station.
```

Origin at Point G



```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 6
At point: G
point: D arrived to charging station.
```

Origin at Point H



```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
Your location: 7
At point: H
point: No pathway towards the nearest charging station.
```

How did the code work?

The code defined the size of the 2d array using a macro first. Since the points are from A to H, we set the length of the column and row of the array to 8. Now that length of the row and column has been declared, the multidimensional array will now be initialized using the bracketed rows and it is in bool type, since the values in the array are 1/0, which is also equivalent to true/false. Once the multidimensional array named *"road_network"* has been set, the code proceeds to display the adjacency matrix.

First, the heading or the first row of points A to H are displayed using the print function and the values in the proceeding rows and columns will be displayed using loops. The first loop determines in which row the loop is iterating (for instance, in the first iteration, it will print 'A' and each iteration has their corresponding values in A to H and it is done using the switch case statement).

After the point in the first column has been displayed, the values (which are 1s or 0s) for the next columns will be displayed using the nested loop. The nested loop determines in which column the loop is iterating. The value to be displayed depends on the adjacency matrix the program has initialized using the nested if-else statement. If a particular [row][column] in the array is true, it will display "1" and if not, "0". Basically, '1' in the matrix means that a certain [row][column] exists.

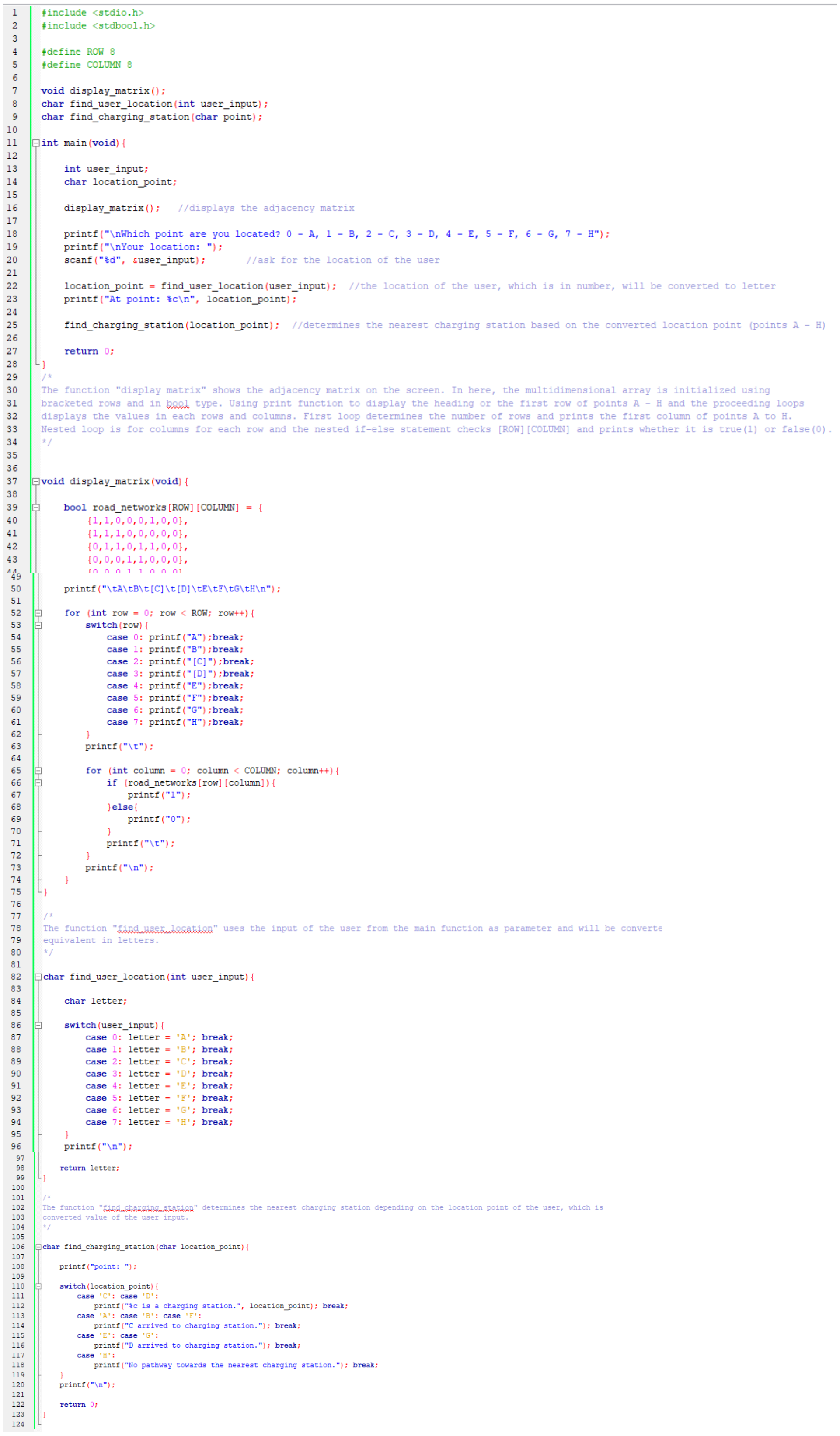
Now that the matrix has been displayed, the program proceeds to ask the user for their location, which accepts integers 0 – 7. Each number corresponds to a point and the program displays its equivalence through the switch case statement. Also, the equivalent will be stored and will be used for the next switch case statement.

Lastly, the code will determine the nearest charging station (which is at point C or D) depending on the user input using the switch case statement as well.

Takeaways / Insights

It was really fun and challenging manipulating the code, especially that each point has certain conditions. Each point has certain pathways leading to another according to the figure shown in the assignment and can be represented by a bool type array. However, I did not apply conditions for each point because I don't want to risk the readability of the code. What I did is pretty straightforward that I displayed the matrix according to the road network figure, ask the user for their location and displays the nearest charging station from that point. I am a programmer that uses minimal lines as possible but delivers the correct outputs to the users. On this assignment, I did code what is being asked to be displayed on the screen but I felt that it is still incomplete because the lessons are focused on arrays specifically. I would like to add more substance, such as indicating that on this point, the available pathways are towards these points and it would be cool if it is also displayed on the screen and updates the user. What I code is merely showing the user his location point and the nearest charging station near it (either C or D). I did not, however, show the flow of pathways from the location or chosen location of the user towards the nearest charging station. That is something I would like to unravel.

Screenshot of the Code: (Using Functions)



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

#define ROW 8
#define COLUMN 8

void display_matrix();
char find_user_location(int user_input);
char find_charging_station(char point);

int main(void){

    int user_input;
    char location_point;

    display_matrix();   //displays the adjacency matrix

    printf("\nWhich point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H");
    printf("\nYour location: ");
    scanf("%d", &user_input);       //ask for the location of the user

    location_point = find_user_location(user_input);  //the location of the user, which is in number, will be converted to letter
    printf("At point: %c\n", location_point);

    find_charging_station(location_point);  //determines the nearest charging station based on the converted location point (points A - H)

    return 0;
}
/*
The function "display matrix" shows the adjacency matrix on the screen. In here, the multidimensional array is initialized using
bracketed rows and in bool type. Using print function to display the heading or the first row of points A - H and the proceeding loops
displays the values in each rows and columns. First loop determines the number of rows and prints the first column of points A to H.
Nested loop is for columns for each row and the nested if-else statement checks [ROW][COLUMN] and prints whether it is true(1) or false(0).
*/


void display_matrix(void){

    bool road_networks[ROW][COLUMN] = {
        {1,1,0,0,0,1,0,0},
        {1,1,1,0,0,0,0,0},
        {0,1,1,0,1,1,0,0},
        {0,0,0,1,1,0,0,0},

    printf("\tA\tB\t[C]\t[D]\tE\tF\tG\tH\n");

    for (int row = 0; row < ROW; row++){
        switch(row){
            case 0: printf("A");break;
            case 1: printf("B");break;
            case 2: printf("[C]");break;
            case 3: printf("[D]");break;
            case 4: printf("E");break;
            case 5: printf("F");break;
            case 6: printf("G");break;
            case 7: printf("H");break;
        }
        printf("\t");

        for (int column = 0; column < COLUMN; column++){
            if (road_networks[row][column]){
                printf("1");
            }else{
                printf("0");
            }
            printf("\t");
        }
        printf("\n");
    }
}

/*
The function "find user location" uses the input of the user from the main function as parameter and will be converte
equivalent in letters.
*/

char find_user_location(int user_input){

    char letter;

    switch(user_input){
        case 0: letter = 'A'; break;
        case 1: letter = 'B'; break;
        case 2: letter = 'C'; break;
        case 3: letter = 'D'; break;
        case 4: letter = 'E'; break;
        case 5: letter = 'F'; break;
        case 6: letter = 'G'; break;
        case 7: letter = 'H'; break;
    }
    printf("\n");

    return letter;
}

/*
The function "find charging station" determines the nearest charging station depending on the location point of the user, which is
converted value of the user input.
*/

char find_charging_station(char location_point){

    printf("point: ");

    switch(location_point){
        case 'C': case 'D':
            printf("%c is a charging station.", location_point); break;
        case 'A': case 'B': case 'F':
            printf("C arrived to charging station."); break;
        case 'E': case 'G':
            printf("D arrived to charging station."); break;
        case 'H':
            printf("No pathway towards the nearest charging station."); break;
    }
    printf("\n");

    return 0;
}
```

GitHub Link: https://github.com/nbbryy/CMSC-21.git