CMSC 21
Lectures 6-7 Assignment

1.

A.

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

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))

int main(){

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

    for (int i = 0; i < NUM_PATHWAYS; i++){

        if (pathway[i]){
            printf("pathway[%d] is open \n", i);

        }else{
            printf("pathway[%d] is close \n", i);
        }
    }
    return 0;
}
```

```
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

Process returned 0 (0x0)   execution time : 0.049 s
Press any key to continue.
```

B.

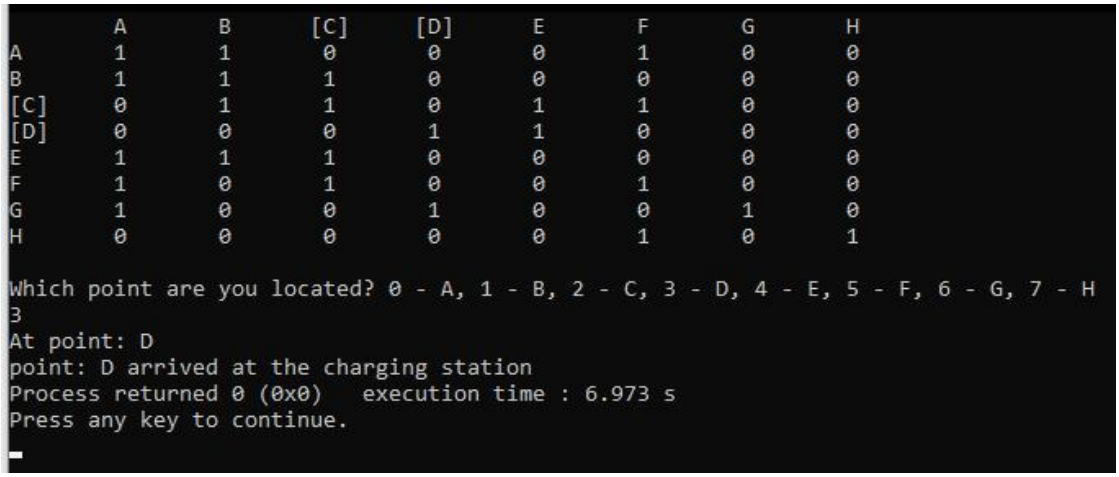
```c
#include <stdio.h>
#include <stdbool.h>

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))

int main(){

    bool pathway[8] = {true, false, true};

    for (int i = 0; i < NUM_PATHWAYS; i++){

        if (pathway[i]){
            printf("pathway[%d] is open \n", i);

        }else{
            printf("pathway[%d] is close \n", i);
        }   int printf(const char*, ...)
    }
    return 0;
}
```

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

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))

int main(){

    bool pathway[8] = {true, 0, true};

    for (int i = 0; i < NUM_PATHWAYS; i++){

        if (pathway[i]){
            printf("pathway[%d] is open \n", i);

        }else{
            printf("pathway[%d] is close \n", i);
        }
    }
    return 0;
}
```

```
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

Process returned 0 (0x0)   execution time : 0.054 s
Press any key to continue.
```

2.

```c
#include<stdio.h>
#define ROWS 8 //* defining variables ROWS and COLUMNS as size 8
#define COLUMNS 8

int main(){ //* calling the main function

    int rows, columns, ans; //* defining variables as integers

    //*assigning an array containing the name of the charging stations into an string array
    char* charging_stations[8] = {"A  " , "B  ", "[C]" ,"[D]", "E  ", "F  ", "G  ", "H  ",};

    //*printing the name of the charging stations on the top
    printf("        A       B       [C]     [D]       E       F       G       H    ");
    printf("\n");

    //*storing all the boolean values of the adjacency matrix into the variable road_networks which is an integer
    int road_networks[ROWS][COLUMNS] = {
                                {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},
                                {1,1,1,0,0,0,0,0},
                                {1,0,1,0,0,1,0,0},
                                {1,0,0,1,0,0,1,0},
                                {0,0,0,0,0,1,0,1},
                                };

    for(rows = 0; rows < ROWS; rows++) //*for loop for displaying the charging stations on the left
    {
        printf("%s   ", charging_stations[rows]);

        for(columns = 0; columns < COLUMNS; columns++) //*for loop displaying the boolean values under each row and column
        {
            printf("  %d     ", road_networks[rows][columns]);
        }
        printf("\n");
    }

     //*asking user for the point he/she is located at
    printf("\nWhich point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H \n");
    scanf("%d", &ans); //*scanning and storing the answer into variable ans

    switch(ans) //*switch case for each charging point
    {
        //*if ans is 0, user is at point A, crossing to point B, then at point C which is the nearest charging station
        case 0:
            printf("At point: A\n");
            printf("Now at point: B");
            printf("point: C arrived at the charging station");
            break;

        case 1: //*if ans is 1, user is at point B, then at point C which is the nearest charging station
             printf("At point: B\n");
             printf("point: C arrived at the charging station");
             break;

         case 2: //*if ans is 2, user is at point C, then at point C which is the nearest charging station
             printf("At point: C\n");
             printf("point: C arrived at the charging station");
             break;

         case 3: //*if ans is 3, user is at point D, then at point D which is the nearest charging station
             printf("At point: D\n");
             printf("point: D arrived at the charging station");
             break;

         case 4: //*if ans is 4, user is at point E, then at point D which is the nearest charging station
             printf("At point: E\n");
             printf("point: D arrived at the charging station");
             break;

         case 5: //*if ans is 5, user is at point F, then at point C which is the nearest charging station
             printf("At point: F\n");
             printf("point: C arrived at the charging station");
             break;

         case 6: //*if ans is 6, user is at point G, then at point D which is the nearest charging station
             printf("At point: G\n");
             printf("point: D arrived at the charging station");
             break;

        /*if ans is 7, user is at point H, then at point C which is the nearest charging station,
        but there is no path towards c from h */
        case 7:
             printf("At point: H\n");
             printf("point: C arrived at the charging station");
             break;
    }
    return 0;
}
```



In the programming assignment, it was difficult to find a solution that was a single algorithm, but in this case, I had to think of a solution which will produce the same result as intended. In the first row where we have to display the points/designations, I use the printf function and used spaces to make it aligned with values in the adjacent matrix. Then, in the adjacent matrix, I stored manually all the boolean values into a variable for the matrix, then printed it with all the necessary spaces to make it as orderly as possible. I had a few problems with printing out the letters A-H in the side, which took me the longest time to do, then the brackets in C and D made it a little more complicated that the spaces have to be adjusted again. Then I used switch case for the second part, where user input is needed, and then the details that I need to include according to the instructions. So far this was the most difficult assignment, and I thought that I have to pass later than the deadline but luckily, I succeeded in making a code that produces an output that is the same as what is required. I hope so.