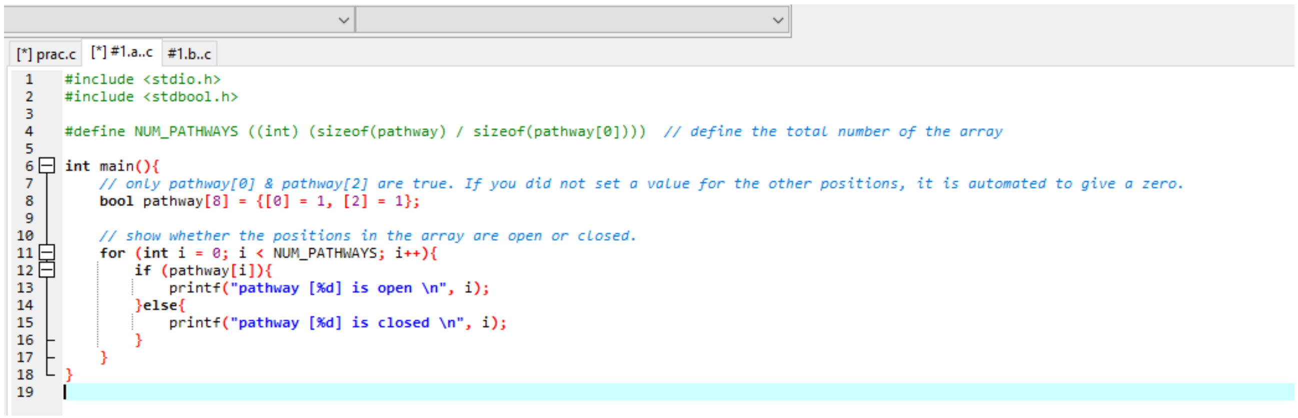
Selena Therese M. Barrameda 202150073

CMSC 21 – 1

Assignment #2 – Lecture 6 & 7

1.

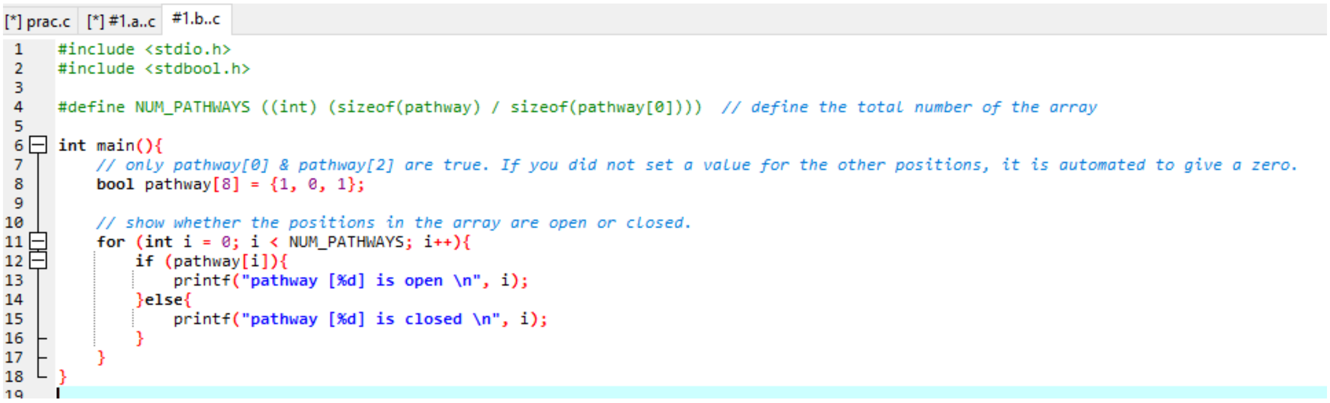
a.



[*] prac.c [*] #1.a..c #1.b..c

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

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))  // define the total number of the array

int main(){
    // only pathway[0] & pathway[2] are true. If you did not set a value for the other positions, it is automated to give a zero.
    bool pathway[8] = {[0] = 1, [2] = 1};

    // show whether the positions in the array are open or closed.
    for (int i = 0; i < NUM_PATHWAYS; i++){
        if (pathway[i]){
            printf("pathway [%d] is open \n", i);
        }else{
            printf("pathway [%d] is closed \n", i);
        }
    }
}
```

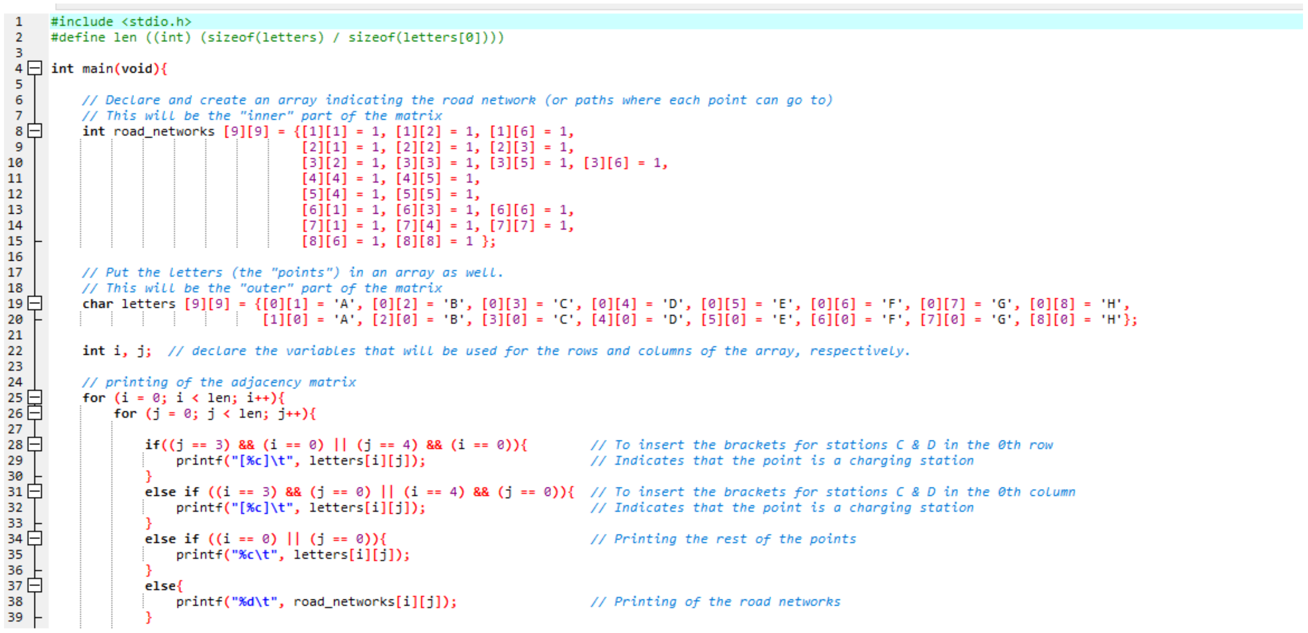
b.



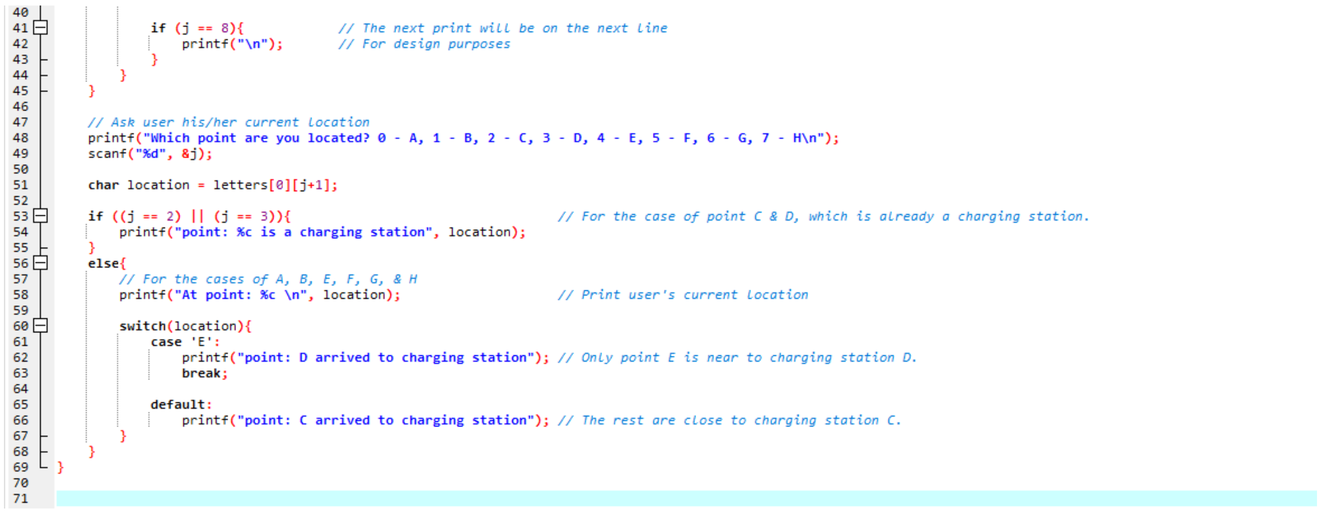
[*] prac.c [*] #1.a..c #1.b..c

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

#define NUM_PATHWAYS ((int) (sizeof(pathway) / sizeof(pathway[0])))  // define the total number of the array

int main(){
    // only pathway[0] & pathway[2] are true. If you did not set a value for the other positions, it is automated to give a zero.
    bool pathway[8] = {1, 0, 1};

    // show whether the positions in the array are open or closed.
    for (int i = 0; i < NUM_PATHWAYS; i++){
        if (pathway[i]){
            printf("pathway [%d] is open \n", i);
        }else{
            printf("pathway [%d] is closed \n", i);
        }
    }
}
```

2.



```c
#include <stdio.h>
#define len ((int) (sizeof(letters) / sizeof(letters[0])))

int main(void){

    // Declare and create an array indicating the road network (or paths where each point can go to)
    // This will be the "inner" part of the matrix
    int road_networks [9][9] = {[1][1] = 1, [1][2] = 1, [1][6] = 1,
                                [2][1] = 1, [2][2] = 1, [2][3] = 1,
                                [3][2] = 1, [3][3] = 1, [3][5] = 1, [3][6] = 1,
                                [4][4] = 1, [4][5] = 1,
                                [5][4] = 1, [5][5] = 1,
                                [6][1] = 1, [6][3] = 1, [6][6] = 1,
                                [7][1] = 1, [7][4] = 1, [7][7] = 1,
                                [8][6] = 1, [8][8] = 1 };

    // Put the letters (the "points") in an array as well.
    // This will be the "outer" part of the matrix
    char letters [9][9] = {[0][1] = 'A', [0][2] = 'B', [0][3] = 'C', [0][4] = 'D', [0][5] = 'E', [0][6] = 'F', [0][7] = 'G', [0][8] = 'H',
                           [1][0] = 'A', [2][0] = 'B', [3][0] = 'C', [4][0] = 'D', [5][0] = 'E', [6][0] = 'F', [7][0] = 'G', [8][0] = 'H'};

    int i, j;  // declare the variables that will be used for the rows and columns of the array, respectively.

    // printing of the adjacency matrix
    for (i = 0; i < len; i++){
        for (j = 0; j < len; j++){

            if((j == 3) && (i == 0) || (j == 4) && (i == 0)){        // To insert the brackets for stations C & D in the 0th row
                printf("[%c]\t", letters[i][j]);                      // Indicates that the point is a charging station
            }
            else if ((i == 3) && (j == 0) || (i == 4) && (j == 0)){  // To insert the brackets for stations C & D in the 0th column
                printf("[%c]\t", letters[i][j]);                      // Indicates that the point is a charging station
            }
            else if ((i == 0) || (j == 0)){                           // Printing the rest of the points
                printf("%c\t", letters[i][j]);
            }
            else{
                printf("%d\t", road_networks[i][j]);                  // Printing of the road networks
            }
```



```c

            if (j == 8){            // The next print will be on the next line
                printf("\n");       // For design purposes
            }
        }
    }

    // Ask user his/her current location
    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", &j);

    char location = letters[0][j+1];

    if ((j == 2) || (j == 3)){                              // For the case of point C & D, which is already a charging station.
        printf("point: %c is a charging station", location);
    }
    else{
        // For the cases of A, B, E, F, G, & H
        printf("At point: %c \n", location);                // Print user's current location

        switch(location){
            case 'E':
                printf("point: D arrived to charging station"); // Only point E is near to charging station D.
                break;

            default:
                printf("point: C arrived to charging station"); // The rest are close to charging station C.
        }
    }
}

```

Because of item #1, I was reminded how to deal with arrays that have boolean as their values. You will only need to specify the positions of the 1s since the rest will be considered as 0s. No need to further type everything, which is really convenient.

The next array is for the “points”. I made an array for this so that it will be easier when coding for the printing of the matrix. If it were not for the array, I might use ifs statements for each letter (which can be a hassle).

A thing that did not work as intended was the paths (where each point can or cannot go to). Even though there is a matrix shown, I was not able to utilize it (the array with boolean values) well when coding for the part where the program tells the user the nearest charging station. Instead, I took note of where each point’s nearing station is. I also almost used only if statements in the printing process (when telling the user the nearest station). It is a good thing that there is a variety of statements (like the switch, which I used in this program) that can be used.

Github link:

https://github.com/selenabarrameda/CMSC-21/tree/main/Lecture%206-7/Assignments