1a. I assign their value fixed through their position.

```
bool pathway[8] = { [0] = 1, [2] = 1};
```

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

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

int main(){

    /*

    A boolean array that contains true/false values referring to
    whether a certain pathway is open/close for transportation.

    Only pathways 0 and 2 are open for transportation. The rest are close.

    */
   bool pathway[8] = { [0] = 1, [2] = 1};

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

       /*

       Displays the status of each pathway.

       Remember that pathway is type bool so its elements are neither true/false
- 1/0.

       */

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

    return 0;
}
```

1b. I assign their value according to Boolean number 1 and 0.

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

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

```c
#include<stdbool.h>

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

int main(){

    /*

    A boolean array that contains true/false values referring to
    whether a certain pathway is open/close for transportation.

    Only pathways 0 and 2 are open for transportation. The rest are close.

    */
   bool pathway[8] = {1, 0, 1};

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

       /*

       Displays the status of each pathway.

       Remember that pathway is type bool so its elements are neither true/false
- 1/0.

       */

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

    return 0;
}
```

2.

Code:

```c
#include <stdio.h>
#include <stdbool.h>
//I made it two to avoid confusion on assigning variables
//but can be fuse as 1.
#define row 8
#define col 8

int main(void)
{
    //initialize road_networks as a bool
    bool road_networks[row][col] =
    {
        {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}
    };
    //for the points/destination of the matrix also initialize new_col_row as a
character
    char new_col_row[8] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
    //initialize loc as an integer
    int loc;
    //inserting the point/destinations in the matrix on column part
    for (int i = 0; i < col; i++)
    {
        if (i != 2 & i != 3)
        {
            printf("\t%c", new_col_row[i]);
        }
        else
        {
            printf("\t[%c]", new_col_row[i]);
        }
    }
    //prints the matrix
    for (int loop_row = 0; loop_row < row; loop_row ++)
    {
```

```
        printf("\n");
        //if else is for the insertion of points/destination in the matrix on the
row part
        if (loop_row != 2 & loop_row != 3) {
            printf("%c", new_col_row[loop_row]);
        } else {
            printf("[%c]", new_col_row[loop_row]);
        }
        //creates the informative matrix
        for (int loop_col = 0; loop_col < col; loop_col++)
        {
            printf("\t%i", road_networks[loop_row][loop_col]);
        }
    }
    //user's input
    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", &loc);
    //initial point
    printf("At point: %c\n", new_col_row[loc]);
    //shows the way to the nearest charging point
    for (int i = 0; i < row; i++)
    {
        if (loc == 2 || loc == 3)
        {
            break;
        }
        else if (loc == 7)
        {
            printf("Now at point: I\n");
            printf("Now at point: %c\n", new_col_row[7]);
            loc = 7;
            break;
        }
        else if(loc == 6)
        {
            printf("Now at point: %c\n", new_col_row[3]);
            loc = 3;
            break;
        }
        else if (road_networks[i][loc] == 1)
        {
            if(loc == i)
            {
                continue;
```

```
            }
            else
            loc = i;
            printf("Now at point: %c\n", new_col_row[i]);
        }
    }

    //prints where are your final destination.
    printf("Point: %c arrived to charging station", new_col_row[loc]);
    return 0;


}
```

Sample result:

```
        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
5
At point: F
Now at point: A
Now at point: B
Now at point: C
Point: C arrived to charging station
```

Explanation:

I will first mention the bonus part, on my case I define row and column into 8 because if they are separate it is easy for me to assign those in the main part if my code, plus it helps me to avoid confusion, but if I set aside my reason I could fuse them as one.

Now for the main part of this assignment, first I made a road matrix by making an 8 arrays with the length of 8, next the instruction says we need to name every point by putting a letter A – H through their column and row, to make that happen I've used a for loop separately for the column but for the row I insert them on another for loop where it would display the road matrix. Next for the point where you are located, I refer to the array I made to display it for the matrix. Next to the destination of the user, what I made is to direct them into the nearest charging point so if the user's initial point is at the point where the charging station, nothing will happen because the code I made is to lead you to the nearest charging point just like for Point G I put a restriction in that point, if you are in point G the only way you have to go is to point D because it is the nearest charging point they don't have to travel to point A anymore, but if you are from

the point F, it will not goes to the nearest charging point which Is to point C because I follow the example from the guide.

4. Bonus: Use a macro to define the size of the 2d array

```
        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
5
At point: F
Now at point A
Now at point B
Now at point C
point: C arrived to charging station
```

I also put a restriction to point H where the only point you could get to is to point I then go back again to point H. in that part on the road matrix the point I doesn't exist but on the other image point I does exist so I include point I there, now for the longer distance I also use a for loop there where they have to refer on the Boolean matrix whether the path is available to reach the nearest charging point so the condition I put there if the value of the path from matrix is 1, you can go there while if it is 0, it means you can't go there.

GITHUB LINK:

https://github.com/JohnLuis07/john/tree/main/CMSC%2021/Lecture%206%20-%207/Assignments