Lecture 6-7 Assignments

1. In the program below, an array named pathway contains eight bool values. Each bool element refers to whether a pathway is open or closed for transportation. Only pathways 0 and 2 are open while the rest are still close due to road constructions and fixings.

    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.

    Specify element 0 and 2 to true, others default to false

a.c b.c

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

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

int main(){
    //bool pathway[8] = {true, false, true, false, false, false, false, false};
    bool pathway[8] = {[0] = true, [2] = true}; //lets element 0 and 2 to true, others default to false
    //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);}
    }
    return 0;
```

"C:\Users\John Hamir Karim\Desktop\CMSC21\Lecture6-7\Assignments\a.exe"

```
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 : 0.048 s
Press any key to continue.
```

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

Start here | a.c | b.c

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

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

int main(){
    //bool pathway[8] = {true, false, true, false, false, false, false, false};
    //bool pathway[8] = {[0] = true, [2] = true};
    bool pathway[8] = {true, false, true}; //Specify value for the first 3 elements, the rest defaults to false

    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;
```

"C:\Users\John Hamir Karim\Desktop\CMSC21\Lecture6-7\Assignments\b.exe"

```
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 : 0.052 s
Press any key to continue.
```

2. In the program below, an array named pathway contains eight bool values.

   As instructed, I created a macro for the size of the adjacency matrix and the multidimensional array itself that represents the matrix.

```c
//define macro int size
#define size 8
int main(){
    //create variables for nodes/points and the adjacency matrix
    char points[9][2] = {"", "A", "B", "C", "D", "E", "F", "G", "H"};
    int road_networks[size][size] = {{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}};
    //print the letters in rows and columns
    printf("      A     B    [C]    [D]    E     F     G     H\n");
```

   To print the row and column of letters for the matrix, I used printf for the row(single line) and for loop+switch..case for the column.

```c
//print the letters in rows and columns
printf("      A     B    [C]    [D]    E     F     G     H\n");
for(int i=0; i<size; i++){
    switch(i){
    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     ");
```

Inside the for loop, I used another loop to print the matrix. This essentially uses the nestep for loop for the matrix.

```
        case 7:
            printf("H      ");
            break;}
        //print the adjacency mattrix
        for(int j=0; j<size; j++){
            if(j==size-1){
                printf("%d      \n", road_networks[i][j]);
            }else{
            printf("%d      ", road_networks[i][j]);
            }
```

For the main operation of the program, we ask the user for the starting point and use a loop to iterate through the rows of 0s and 1s from the starting point.

```
//variable for the user choice of starting point
int starting_point;
printf("Which point are you located?: 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H");
printf("\nAt point: ");
scanf("%d", &starting_point);
printf("Starting at point: %c\n", points[starting_point][2]);

//let i = starting point
int i=starting_point;
    for(int j=0;j<size;j++){ //iterate through all possible routes
        if(i==j){ //skip the route with the same letter
            continue;
            }
        else if(i==2 || i==3){//stop loop when at C or D
            break;
            }
        else if(road_networks[i][j]==1){//find the immediate route and change the value used for starting point
            printf("Now at point: %c\n", points[j][2]);
            i=j;
            }
        }
    printf("Arrived at point: %c charging station", points[i][2]);//print final destination
```

Since the same letter path is always = 1, we try to skip it through a conditional(line 63). However, this method fails when a 1 is found before the chosen starting point.

We also terminate the loop if the user is already at the charging station(i==2 and i==3).

The last conditional finds a value of 1 in the row, prints the letter where the 1 was found, then changes the value of the starting point. This then repeats the loop to find the closest 1 again.

If the starting point ends up either 2 or 3, the loop terminates and the program prints the destination(charging station) before terminating altogether.