Precious Kaira B. Saluria

202101275

Lecture 607 Assignment

1.

a)

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

b)

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

2.

```
/*****************************
 *  SALURIA, PRECIOUS KAIRA   *
 * LECTURE 6-7 ASSIGNMENT #2  *
 *****************************/

#include <stdio.h>

#define ROW 8
#define COLUMN 8

// FUNCTION FOR PRINTING THE ADJACENCY MATRIX
void printing(char letter [8], int matrix[ROW][COLUMN]){

    int i=0, l=0;
    for(i; i < ROW; i++) {

        // printing the station labels in every row
        if (i == 2 || i == 3){          //  if the letter is a charging station
            printf ("[%c]", letter[l]);  // adds a bracket
            l++;
        }
        else{
            printf ("%c", letter[l]);
            l++;
        }

        // iterates through each element of the array and print it
```

```
        for (int j=0; j<8; j++) {
        printf("\t%d ", matrix[i][j]);
        }
        printf("\n");

    }
}

// FUNCTION TO DETERMINE THE NEAREST CHARGING STATION GIVEN A LOCATION
void measure( int station, int matrix[ROW][COLUMN], char letter[8]){

    // if the initial locatin is already a charging station
    if (station == 2 || station == 3){
        printf ("Point %c is already a charging station!", letter[station]);
    }
    // if there is a direct path between the location and charging station C
    else if (matrix[station][2] == 1){
        printf ("Now at point C\n");
        printf ("Arrived at the nearest charging station!");
    }
    // if there is a direct path between the location and charging station D
    else if (matrix[station][3] == 1){
        printf ("Now at point D\n");
        printf ("Arrived at a charging station!");
    }
    // if there is no direct path between the point and any of the two charging
station
    else{

        // to iterate the columns starting from index 0
        for (int new_station = 0; new_station < COLUMN; new_station++){

        // if the point is equal to 1 and is not the starting point
        if ((matrix[station][new_station]==1) && (new_station!=station) ){
            printf ("Now at point %c\n", letter[new_station]);  // print the
connecting path
            station = new_station;  // index of that point will be the new
starting point

            measure(station, matrix, letter);  // check the new point
            break;

            }
        }
    }
```

```c
}

int main (){

    int location;

    // ARRAY INITIALIZATION AND DECLARATION
    int road_networks [ROW] [COLUMN] = {
        {1, 1, 0, 0, 0, 1, 0, 0},  // A
        {1, 1, 1, 0, 0, 0, 0, 0},  // B
        {0, 1, 1, 0, 1, 1, 0, 0},  // [C]
        {0, 0, 0, 1, 1, 0, 0, 0},  // [D]
        {0, 0, 0, 1, 1, 0, 0, 0},  // E
        {1, 0, 1, 0, 0, 1, 0, 0},  // F
        {1, 0, 0, 1, 0, 0, 1, 0},  // G
        {0, 0, 0, 0, 0, 1, 0, 1},  // H
        };

    char station_letters [8] = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};

    // PRINTING THE LETTERS IN EVERY COLUMN
    printf ("\tA\tB\t[C]\t[D]\tE\tF\tG\tH\n");

    // CALLING OUT FUNCTION TO DISPLAY THE ADJACENCY MATRIX
    printing(station_letters, road_networks);

    // POINTS/STATIONS
    printf ("\nWhich point are you located?\n\n  [0 - A] \t [4 - E]\n  [1 - B] \t [5 - F]\n  [2 - C] \t [6 - G]\n  [3 - D] \t [7 - H]\n");

    // INPUT VALIDATION FOR LOCATION
    do{
        printf ("\nEnter current location (0-7): ");
        scanf ("%d", &location);

        if (location < 0 || location > 7){
            printf ("Invalid input! Try again.\n");
        }
    } while (location < 0 || location > 7);

    printf ("\nAt point: %c\n", station_letters[location]);

    // CALLING OUT FUNCTION TO DETERMINE THE NEAREST CHARGING STATION
    measure(location, road_networks, station_letters);
```

```
    return 0;
}
```

The first thing I did is to declare and initialize a road _networks multidimensional array that will represent the adjacency matrix. I defined a macro with the help of the #define preprocessor directive to define the size of the 2d array.

To print the adjacency matrix, I used the printf function to print the letters for every column of the matrix. Then I created a function named printing to print the values in the road_network array. The block of code inside the main for loop in the printing function will be continuously executed until the test expression is evaluated to true. The if-else statement in the for loop is for printing the station labels in every row while the nested for loop iterates through each element of the road_networks array and prints it.

The user is then asked at which point he/she is located, and the input is stored in the location variable. For input validation, I used a do-while loop to make sure that the input entered by the user is between 0-7 only.

To determine the nearest charging station, I defined another function named measure. If the initial location is already a charging location, then the statements inside the if statement is executed. The first else if statement checks if there is a direct path between the location and charging station C. The second else checks if there is a direct path between the location and charging station D. However, if there is no direct path between the location and any of the two-charging stations, then we look for a connecting path and that will be the new location. We then check if that new location has a direct path towards the charging station. If none, then we search for another connecting path and do the same process until we arrive at a charging station.