Franz Peter A. Ferrer

Array/Multi-Dimensional Array

Lecture 4 Assignments

1.
   a.

```
/*****************************
**  Franz Peter A. Ferrer   **
**  Lecture 6-7 Assignment  **
*****************************/

#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 reffering to
    whether a certain pathway is open/close for transportation.

    Only pathways 0 and 3 are open for transportation. The rest are close.

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

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

       /*

       Display the status of each pathway.

       Remember  that pathway is type of bool so its elements are either
true/false - 1/0.

       */

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

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

    return 0;
}
```

b.

```
/*****************************
**  Franz Peter A. Ferrer   **
**  Lecture 6-7 Assignment  **
*****************************/
#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 reffering to
    whether a certain pathway is open/close for transportation.

    Only pathways 0 and 3 are open for transportation. The rest are close.

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

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

        /*

        Display the status of each pathway.

        Remember  that pathway is type of bool so its elements are either
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.

```
/*****************************
**  Franz Peter A. Ferrer   **
**  Lecture 6-7 Assignment  **
*****************************/
#include <stdio.h>

#define ARR_SIZE 8 //macro for the size(rows and columns) of the array

/*Void Function - Logic for the result
Note that the charging station are station C(2) and D(3)
This functions is utilized in order to find what is the nearest charging
stations*/
void path_alg(int nodes[ARR_SIZE][ARR_SIZE], int start, char stations[8]);
void path_alg(int nodes[ARR_SIZE][ARR_SIZE], int start, char stations[8]){
    /*If user input is 2 or 3, prints
    a statement that the chosen starting
    point is already a charging station*/
    if (start == 2 || start == 3){
        printf("Point: %c is a charging station.", stations[start]);
    }
    /*If the expression - nodes[start][2] is equal to 1
    This means that the chosen starting point has a direct
    path to the charging station
C.*/
    else if (nodes[start][2] == 1){
        printf("Point: C arrived to charging station.");
    }
    /*Else if the expression - nodes[start][3] is equal to 1
    This means that the chosen starting point has a direct
    path to the charging station D.*/
    else if (nodes[start][3] == 1){
        printf("Point: D arrived to charging station.");
    }
    /*If there is no direct path from the chosen
    starting point to the charging stations.
    Finds an element of array that has a direct path
    to another point*/
```

```
    else {
        /*Iteration for the column, in order to find a path to go to the next
point*/
        for(int column = 0; column < ARR_SIZE; column++){
            /*If the current element of the array is 1,
            Prints the current point and then sets the row
            of the index, from the column. It will now be
            the new starting point. Then, the function will
            be called again in order to check the current point.*/
            if(nodes[start][column] == 1 && column != start){
                printf ("At point: %c\n", stations[column]);
                start = column;
                path_alg(nodes, start, stations);
                break;
            }
        }
    }
}

/*Main Function*/
int main(){

    // Variable Declaration and Initialization
    int start;
    char stations[] = {'A','B','C','D','E','F','G','H'};
    int road_networks[ARR_SIZE][ARR_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}
    };

    //For Loop - Printing the nodes(horizontally) which are the
points/desinations
    for(int letters = 0; letters < ARR_SIZE; letters++){
        /*Since 2 is C and 3 is D, it is put inside a bracket
        which are considered as charging stations*/
        if (letters == 2 || letters == 3){
            printf("\t[%c]", stations[letters]);
        }
        else{
```

```
            printf("\t%c", stations[letters]);
        }
    }
    printf("\n");

    // For loop - Printing if there is a direct path between the two points and
the nodes(vertically)
    for (int letters = 0, i = 0; letters< ARR_SIZE; letters++, i++){
        if (letters == 2 || letters == 3){
            printf("[%c]", stations[letters]);
            for(int j = 0; j < ARR_SIZE; j++){
                printf("\t%d", road_networks[i][j]);
            }
        }
        else{
            printf("%c", stations[letters]);
            for(int j = 0; j < ARR_SIZE; j++){
                printf("\t%d", road_networks[i][j]);
                }
        }
        printf("\n");
    }

    //USER INPUT
    printf("Which point are you located? ");
    printf("0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H\n");
    scanf("%d", &start);
    printf("At point: %c\n", stations[start]);

    //FUNCTION CALL
    path_alg(road_networks, start, stations);

    return 0;
}
```

The function of this program is to determine the nearest charging station, given a point of origin. I used a separate function for the path algorithm which functions to find the nearest charging station. I used if-else-if statements so that I can set point C and point D as the charging station. Next, an else-if statement where it determines if the point of origin has a direct path to Point C or Point D. If the expression has the element 1, it means that it has a direct path to the next point. Then, an else statement where if there is no direct path from the point of origin to a charging station, it will traverse that row until it finds a column of a node or expression that has the element 1. Then it will set the value of that

column into start and will call the function. The function will now be executed again and used the start value as the point of origin.

For the main function, I have the variable declaration, as well as the initialization of the array. I used the macro as the dimension of the array since it has 8 columns and 8 rows. In printing or displaying the matrix, I simply used the for loop. I also used if statement inside the for loop, this is utilized so that I can print the station C and D inside a bracket, so that it can be identified that they're the charging stations. Then, I used scan function for the user-input, this will be the start or point of origin. Lastly, is the function call to execute the path algorithm function.