Cordero, Palmsdale Kevin D.
2021-03842
pdcordero@up.edu.ph

Lectures 6-7 Assignment

1a.

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

1b.

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

2.

```
/* ~~~~~~~~~~~~~~~~~~~~~~~~
Cordero, Palmsdale Kevin D.
2021-03842
pdcordero@up.edu.ph
~~~~~~~~~~~~~~~~~~~~~~~~ */

#include <stdio.h>

#define ROW 8
#define COL 8

/*~~~ GLOBAL VARIABLES ~~~*/
char points[9] = "ABCDEFGH";

int road_networks[ROW][COL] = {
    //       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},
};

/*~~~ FUNCTIONS ~~~*/
void print_network (void) {
    /* Prints out the entire road network */
    int i, j;

    // print point letters with corresponding spacing and inclusion of bracket
    printf("       ");
    for (i = 0; i < ROW; i++) {
```

```c
        switch (i) {
            case 1:
                printf("%-5c", points[i]);
                break;

            case 2:
                printf("[%c]   ", points[i]);
                break;

            case 3:
                printf("[%c]    ", points[i]);
                break;

            default:
                printf("%-6c", points[i]);
                break;
        }
    }

    // print each row with their road_network values
    for (i = 0; i < ROW; i++) {
        // for row header
        if (i==2 || i==3) {
            // for charging stations, input bracket
            printf("\n[%c]    ", points[i]);
        } else {
            printf("\n %-6c", points[i]);
        }

        // for road_network values
        for (j = 0; j < COL; j++) {
            printf("%-6d", road_networks[i][j]);
        }
    }
}

int valid_inp (void) {
    /* Ensures user input is within valid range of 0 to 7 */
    int cur_point;

    while(1) {      // only ends when user inputs valid input
        printf("\nInput [0-7]: ");
        scanf("%d", &cur_point);

        if (cur_point<0 || cur_point>7) {
            printf("Invalid input! Must be integer from 0 to 7. Try again.");
        } else {
            return cur_point;
        }
    }
}

void traverse_to (int cur_point) {
    /* Traverses the current point until it reaches a charging station */
    printf("At point %c\n", points[cur_point]);

    if (cur_point==2 || cur_point==3) {
        // cur_point is at a charging station, end traverse_to recursive
function, print arrival
        printf("Arrived at charging station %c\n", points[cur_point]);

    } else if (road_networks[cur_point][2]) {
        // cur_point can go to charging station C, traverse there
        printf("==> ");
        traverse_to(2);

    } else if (road_networks[cur_point][3]) {
        // cur_point can go to charging station D, traverse there
        printf("==> ");
        traverse_to(3);

    } else {
        // cur_point is no where immediate to a charging station
        int nxt_point;

        for (nxt_point = 0; nxt_point < 8; nxt_point++) {
            // find traversable point, traverse there
            if ((nxt_point != cur_point) &&
(road_networks[cur_point][nxt_point])) {
                printf("==> ");
                traverse_to(nxt_point);
                break;
            }
        }
    }
}

int main (void) {
    int cur_point;
    print_network();

    printf("\n\nSet initial point:\n    (0) A\n    (1) B\n    (2) C\n    (3)
D\n    (4) E\n    (5) F\n    (6) G\n    (7) H");
    cur_point = valid_inp();

    printf("\n");
    traverse_to(cur_point);

    return 0;
}
```

To discuss this, let’s follow the flow of the code.

So first, we used macro to define the dimensions of the 2d array that is ROW and COL at 8 since the matrix is 8 by 8. Then, we declared our global variables. These are char points[9] which is the array for the names of the points, then int road_networks[ROW][COL] which is the array for point adjacency. For road_networks, I used comments to label the matrix rows and columns, making it easier for readers to understand the adjacency matrix.

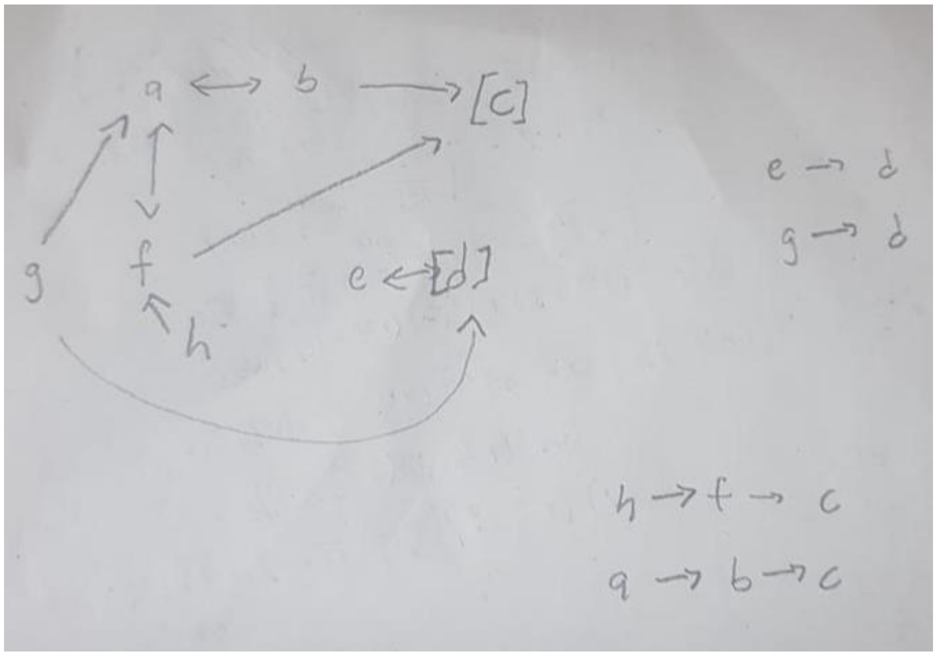
Then we are at the main function, we mostly just call the user-defined functions here. I did it like this so that I could compartmentalize the code into different chunks, making code work easier and more organized.

First user defined function is void print_network(void) where we just print the adjacency matrix with corresponding spacings and brackets. It’s a void function since we don’t need a returned value. It uses for-loops to traverse over the arrays and print the needed values.

Then, we print the choices where the user can choose to start from. This will be assigned as our cur_point. Since there are specific values needed, which are 0-7, we have the int valid_inp(void) function to ensure that cur_point is within the valid range. It uses a while-loop that only ends when returning a valid input, otherwise it will prompt the user again to input.

Finally, we will now traverse point to point using the void traverse_to (int cur_point) function. It is a void function since we only need to move, no returned value. In here, we will print our current location which is the cur_point. Then, we check if the cur_point is at a charging station; if so, we print that we have arrived and thus end the whole program. Else, we now need to move our cur_point to a next point. We first check if the cur_point is adjacent/traversable to a charging station; if so, traverse there. Else, we just go to traverse to a new point. It must not be itself and is adjacent to cur_point based on road_networks; if so, we just go to that point. This will happen until we arrive at charging stations C or D. With this function, we also could count the number of movements needed by counting the amount of printed “==>” arrows. It also traverses with the least number of needed movements based on the given adjacency matrix from the Assignment pdf file.

This was the algorithm I made because the given network system was simple enough. I made a diagram like this to understand the needed traversing or movement from point to point until arriving at a charging station:


a
b
[c]
g
f
e
[d]
h
e → d
g → d
h → f → c
a → b → c