John Es' Ven Britanico (2022-02630)

Loops and Arrays Lecture 4 Assignments

1.

```c
#include <stdio.h>

//1. What is the output of the following program?

/* This code displays powers of two from 1 to 128. The variable i is first initialized to 1, and after that, a while loop is started, which will continue to run as long as i is less than or equal to 128.
The current value of i is first printed using printf within the loop, and then it is multiplied by 2 to form the following power of 2. Until i exceeds 128, this process is repeated. */

int main(void)
{
  int i;

  i = 1;
  while (i <= 128) {
    printf("%d ", i);
    i *= 2;
  }


  return 0;
}

/* This happens because it prints 1 first, then multiplies i by 2 to obtain 2, prints that, then multiplies i by 2 again to get 4, and so on until it prints 128, which is the last power of 2 that is less than or equal to 128. */

//Output: 1 2 4 8 16 32 64 128

//By: John Es' Ven Britanico
```

2.

```c
#include <stdio.h>


//2. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?


int main(void)
{
  /* Assuming that the loop bodies are the same, we generate the following code block: */

  int i;


  printf("\na) while loop:\n");
  i = 1;
  while (i < 10)
  {
    printf("%d ", i);
    i++;
  }

  printf("\n");


  printf("\nb) for loop:\n");
  i = 1;
  for (; i < 10;)
  {
    printf("%d ", i);
    i++;
  }

  printf("\n");


  printf("\nc) do-while loop:\n");
  i = 1;
  do
  {
    printf("%d ", i);
    i++;
  } while (i < 10);

  printf("\n");


  return 0;
}

/*
Output:
a) while loop:
1 2 3 4 5 6 7 8 9

b) for loop:
1 2 3 4 5 6 7 8 9

c) do-while loop:
1 2 3 4 5 6 7 8 9
*/

/* When the first and third expressions are both omitted, the resulting loop
is nothing more than a while statement in disguise. */

//Answer: c

/* The do-while loop in statement (c) ensures that the loop body will run at least once even if the loop condition is false from the start, which is the key difference. while the for loop in statement (b) and the while loop in statement (a) will both never execute the body of the loop if the loop condition is false from the start.

Since the do while loop in statement (c) behaves differently than the other two statements, assuming that the loop bodies are the same, statement (c) is not equivalent to the other two statements. */


//By: John Es' Ven Britanico
```

3.

```c
#include <stdio.h>

//3. Convert item 1 into an equivalent for statement. You can validate your answer by checking if the produced outputs by both the while and for statements are similar.

int main(void)
{
    int i;

    for (i = 1; i <= 128; i *= 2) {
        printf("%d ", i);
    }


    return 0;
}

/* While using a different syntax, the for loop contains the same loop condition and loop body as the while loop. The loop starts with a value of 1 for the variable i,
then iterates until i is less than or equal to 128 by multiplying i by 2 each time. The program prints a series of powers of 2 in the same order as the while loop. */

//Output: 1 2 4 8 16 32 64 128

//By: John Es' Ven Britanico
```

4.

```c
//Includes the standard input and output library that contains input-output functions.
#include <stdio.h>

/*
4. Write a code that computes for the power of two:

TABLE OF POWERS OF TWO
n           2 to the n
---     ---------------
0             1
1             2
2             4
3             8
4             16
5             32
6             64
7             128
8             256
9             512
10            1024
*/


//Defines the main function of the program.
int main()
{

    printf("TABLE OF POWERS OF TWO\n");
    printf("n\t2 to the n\n");
    printf("---\t---------------\n");

    //Declare a variable called "power" and initializes it to 1. This variable will be used to store the current power of 2.
    int power = 1;

    //This line starts a loop that will run 11 times, with "n" starting at 0 and increasing by 1 each time until it reaches 10. This loop will print the powers of 2 from 2^0 to 2^10.
    for (int n = 0; n <= 10; n++) {
        //This line prints the current value of "n" and "power" to the console. The "\t" creates a horizontal space between the two values, and "\n" creates a new line after the message.
        printf("%d\t%d\n", n, power);
        //This line doubles the value of "power", so that the next iteration of the loop will print the next power of 2.
        power *= 2;
    }


    //The program returns 0 to indicate successful execution.
    return 0;
}



//By: John Es' Ven Britanico
```

5.

```c
//Includes the standard input and output library that contains input-output functions.
#include <stdio.h>


/* Write a program that displays a one-month calendar.
There should be a user prompt to set:
• The number of days
• The day of the week on which the month begins.
Additionally, add checkers to validate whether the days entered are valid. For instance,
the following number of days are invalid: 32, -1, 0, 27.
This addition will be a good refresher to our previous topic, selection statements.  */


//Defines the main function of the program.
int main()
{
    //This line declares four integer variables
    int days, starting_day, i, j;

    // Get input from user
    printf("Enter the number of days in the month (28-31): ");
    scanf("%d", &days);

    //Checks if the value entered for days is less than 28 or greater than 31. If so, it prints an error message and exits the program with a status code of 1.
    if (days < 28 || days > 31) {
        printf("Invalid number of days.\n");
        return 1;
    }

    printf("Enter the starting day of the week (1=Sun, 2=Mon, ..., 7=Sat): ");
    scanf("%d", &starting_day);

    //Checks if the value entered is less than 1 or greater than 7. If so, it prints an error message and exits the program with a status code of 1.
    if (starting_day < 1 || starting_day > 7) {
        printf("Invalid starting day.\n");
        return 1;
    }

    // Print calendar header
    printf("\n---------------------------------");
    printf("\nSun  Mon  Tue  Wed  Thu  Fri  Sat\n");

    //This for loop prints out a number of blank spaces at the beginning of the calendar, depending on the starting day of the week.
    //It does this by looping through a certain number of times (determined by the value of (starting_day + 6) % 7) and printing out five spaces for each iteration.
    for (i = 0; i < (starting_day + 6) % 7; i++) {
        printf("     ");
    }

    // Print days
    for (j = 1; j <= days; j++) {
        printf("%-4d ", j);

        // If this is the last day of the week, start a new line
        if ((j + starting_day - 1) % 7 == 0) {
            printf("\n");
        }
    }

    // Print calendar footer
    printf("\n");
    printf("---------------------------------\n");


    //The program returns 0 to indicate successful execution.
    return 0;
}



//By: John Es' Ven Britanico
```

6.

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

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

int main(void){

  /*

  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 closed.

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

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

    /*

    Display the status of each pathway.

    Remember that pathway is type 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 closed \n", i);
    }
  }

  return 0;
}

// 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.

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



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

//Answer:
bool pathway[8] = {true, false, true};
//The remaining elements in the array will be automatically initialized to false.



//By: John Es' Ven Britanico
```

7.

```c
#include <stdio.h>

//Macro to define the size of the 2d array
#define ROWS 9
#define COLS 9


int main() {
    int road_networks[ROWS][COLS] = {
        {1, 1, 0, 0, 0, 1, 0, 0, 0},
        {1, 1, 1, 0, 0, 0, 0, 0, 0},
        {0, 1, 1, 0, 1, 1, 0, 0, 1},
        {0, 0, 0, 1, 1, 0, 0, 0, 0},
        {0, 0, 0, 1, 1, 0, 0, 0, 0},
        {1, 0, 1, 0, 0, 1, 0, 0, 0},
        {1, 0, 0, 1, 0, 0, 1, 0, 0},
        {0, 0, 0, 0, 0, 1, 0, 1, 1},
        {0, 0, 0, 0, 0, 0, 0, 1, 1}
    };

    printf("\n");
    // Display the adjacency matrix
    printf("  a b c d e f g h i\n");
    for (int i = 0; i < ROWS; i++) {
        printf("%c ", 'a' + i); // Print the point/destination label
        for (int j = 0; j < COLS; j++) {
            if (road_networks[i][j]) {
                printf("1 "); // Print 1 if there's a direct path
            } else {
                printf("0 "); // Print 0 if there's no direct path
            }
        }


        printf("\n");
    }

    //Put a bracket to the points/destinations that are considered as charging stations

    printf("\n");

    printf("The adjacency matrix:\n");
    printf("   a   b   c   d   e   f   g   h   i\n");
    printf("a  1   1  [0] [0]  0   1   0   0   0\n");
    printf("b  1   1  [1] [0]  0   0   0   0   0\n");
    printf("c [0] [1] [1] [0] [1] [1] [0] [0] [1]\n");
    printf("d [0] [0] [0] [1] [1] [0] [0] [0] [0]\n");
    printf("e  0   0  [0] [1]  1   0   0   0   0\n");
    printf("f  1   0  [1] [0]  0   1   0   0   0\n");
    printf("g  1   0  [0] [1]  0   0   1   0   0\n");
    printf("h  0   0  [0] [0]  0   1   0   1   1\n");
    printf("i  0   0  [0] [0]  0   0   0   1   1\n");

    printf("\n");

    //Given a point/destination, determine the nearest charging station
    int charging_stations[] = {2, 3};

    int current_location, nearest_charging_station;
    printf("Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H, 8 - I\n");
    scanf("%d", &current_location);


    if (current_location < 0 || current_location >= ROWS) {
        printf("Error: Invalid location\n");
        return 1;
    }

    // Check if current location is a charging station or no path
    if (current_location == 2) {
        printf("point: C is a charging station\n");
        return 0;
    } else if (current_location == 3) {
        printf("point: D is a charging station\n");
        return 0;
    } else if (current_location == 7) {
        printf("At point: H\nThere is no path for charging station\n");
        return 0;
    } else if (current_location == 8)
    {
        printf("At point: I\nThere is no path for charging station\n");
        return 0;
    }



    // Find the nearest charging station
    nearest_charging_station = -1;
    int min_distance = ROWS + COLS;
    for (int i = 0; i < sizeof(charging_stations)/sizeof(int); i++) {
        int station = charging_stations[i];
        if (road_networks[current_location][station] == 1) {
            nearest_charging_station = station;
            break;
        }
        for (int j = 0; j < ROWS; j++) {
            if (road_networks[current_location][j] == 1 && road_networks[j][station] == 1) {
                int distance = j - current_location;
                if (distance < 0) {
                    distance = -distance;
                }
                if (distance < min_distance) {
                    min_distance = distance;
                    nearest_charging_station = station;
                }
            }
        }
    }

    if (nearest_charging_station != -1) {
        printf("At point: %c\npoint: %c Arrived to charging station\n", current_location + 'A', nearest_charging_station + 'A');
    } else {
        printf("At point: %c\npoint: %c is a charging station\n", current_location + 'A', current_location + 'A');
    }

    return 0;
}


//By: John Es' Ven Britanico
```