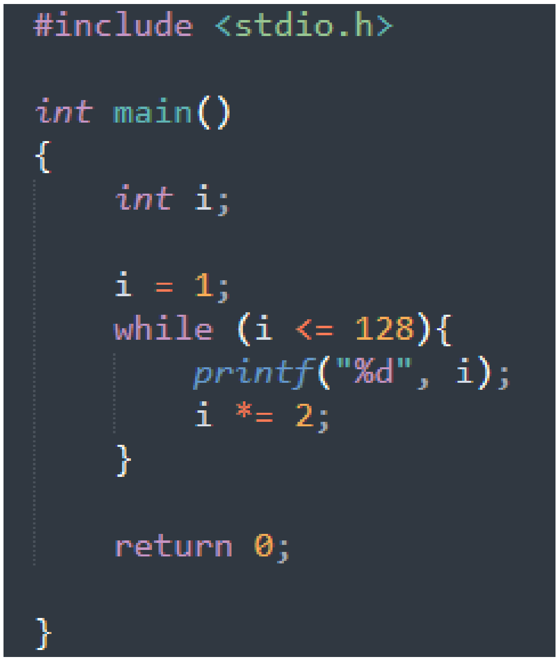
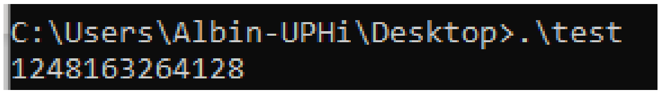
Albin Alexis A. Cayanan CMSC 21 – 1
2022 – 13990

Loops and Arrays
Lecture 4 Assignments

1. What is the output of the following program?



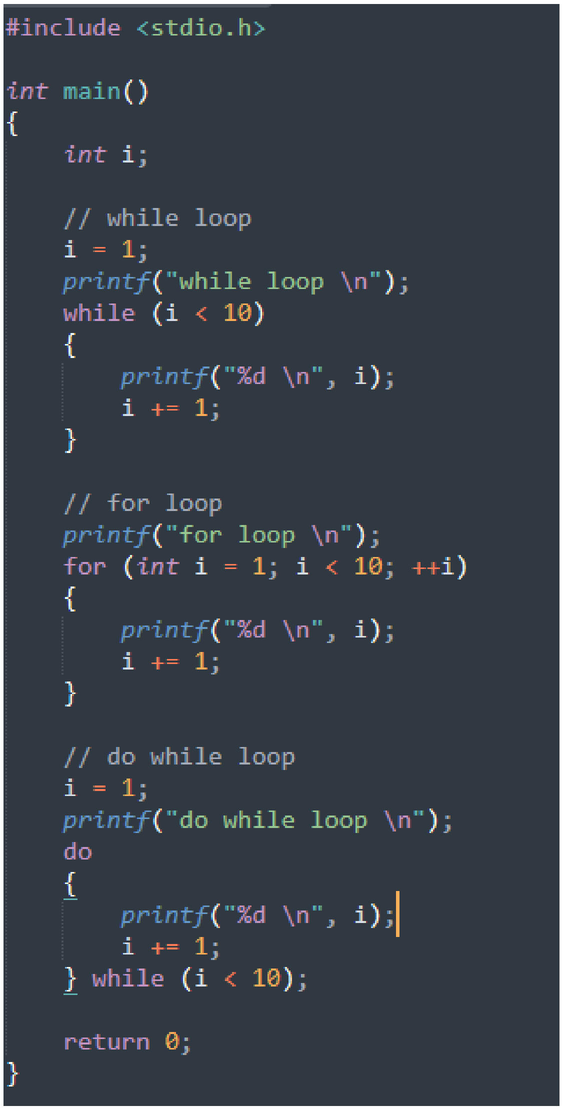
```c
#include <stdio.h>

int main()
{
    int i;

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

    return 0;

}
```



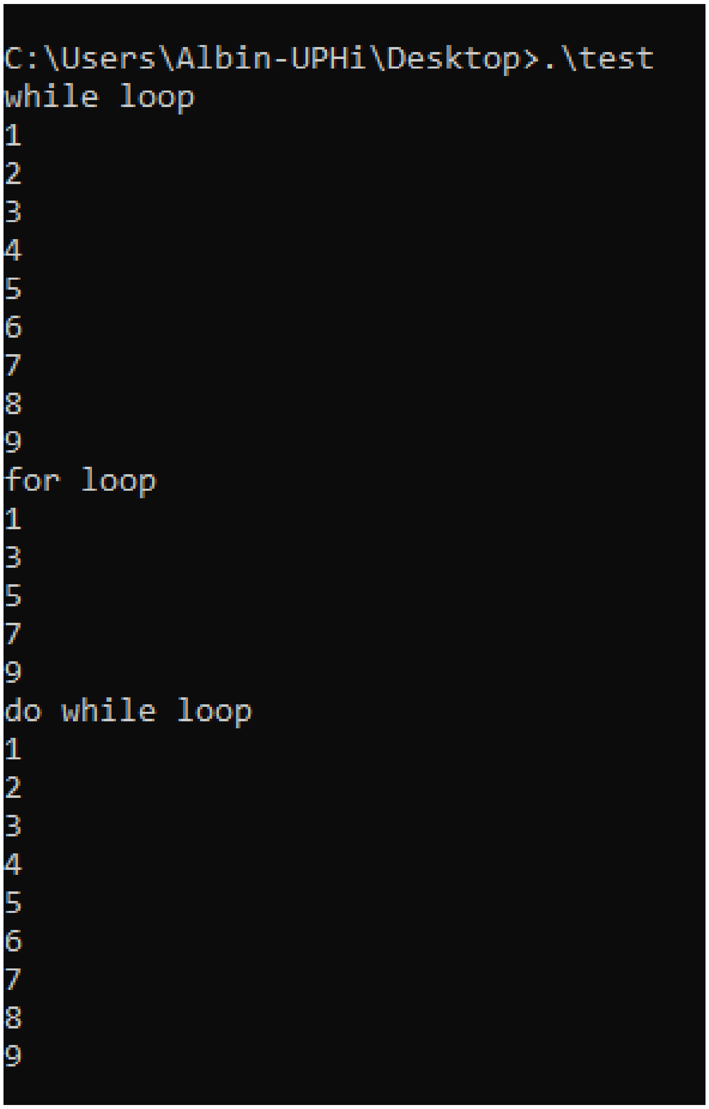
```
C:\Users\Albin-UPHi\Desktop>.\test
1248163264128
```

The output of the program is 1248163264128.

2. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?
   a. `while (i < 10) {...}`
   b. `for (; i < 10;) {...}`
   c. `do {...} while (i < 10;`



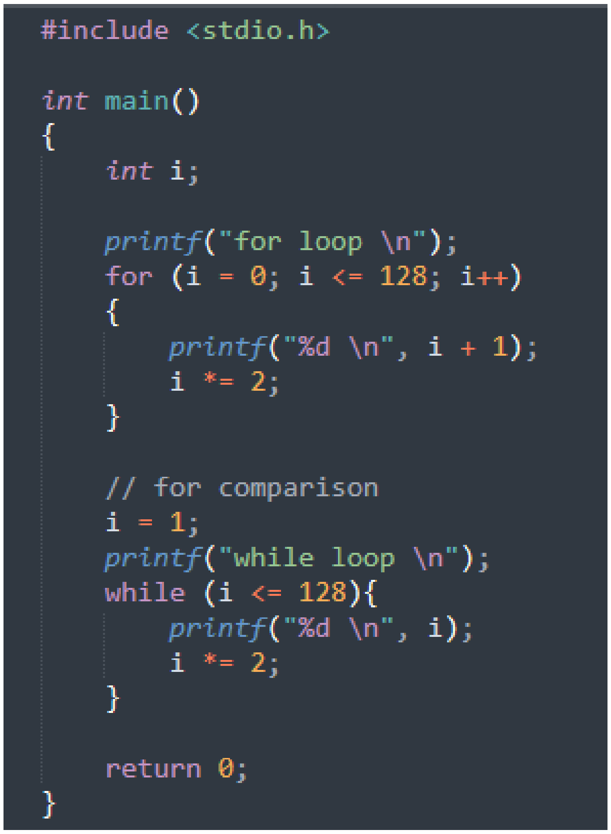
```c
#include <stdio.h>

int main()
{
    int i;

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

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

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

    return 0;
}
```



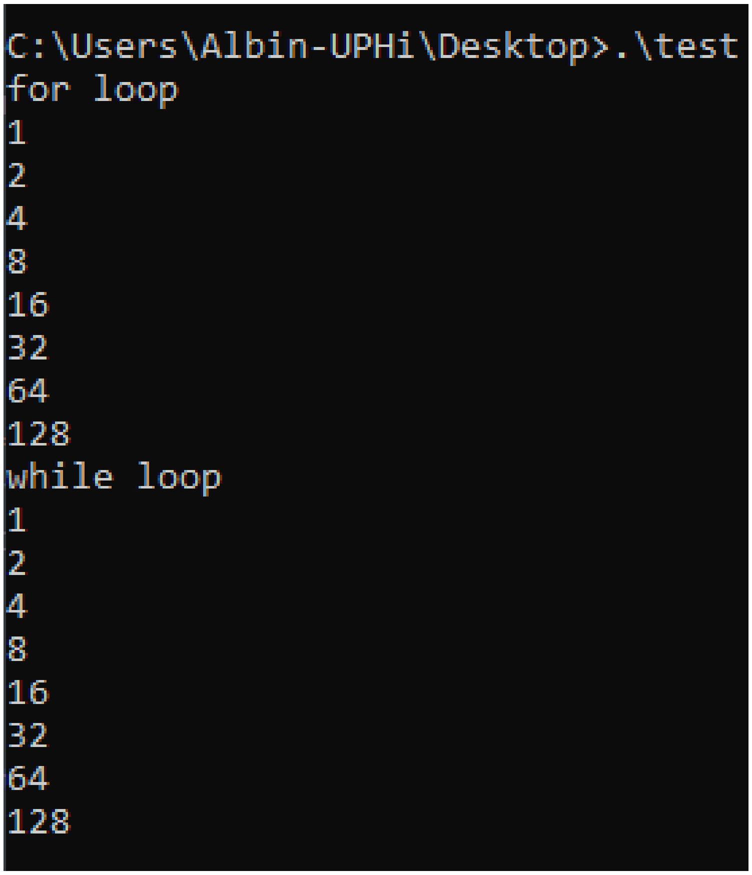
```
C:\Users\Albin-UPHi\Desktop>.\test
while loop
1
2
3
4
5
6
7
8
9
for loop
1
3
5
7
9
do while loop
1
2
3
4
5
6
7
8
9
```

Statement b is not equivalent to statement a and c.

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.



```c
#include <stdio.h>

int main()
{
    int i;

    printf("for loop \n");
    for (i = 0; i <= 128; i++)
    {
        printf("%d \n", i + 1);
        i *= 2;
    }

    // for comparison
    i = 1;
    printf("while loop \n");
    while (i <= 128){
        printf("%d \n", i);
        i *= 2;
    }

    return 0;
}
```

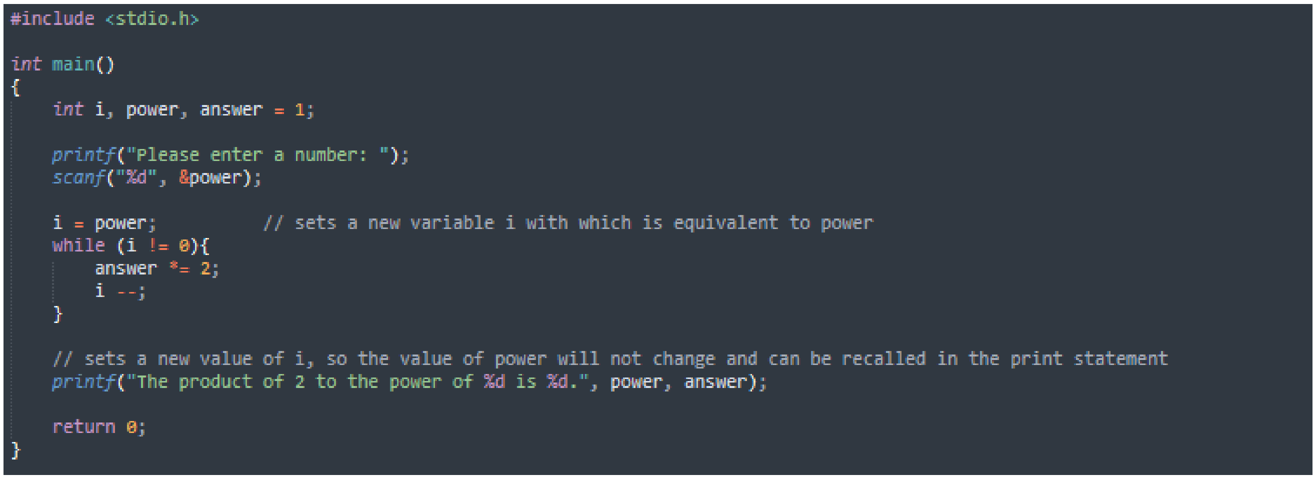


```
C:\Users\Albin-UPHi\Desktop>.\test
for loop
1
2
4
8
16
32
64
128
while loop
1
2
4
8
16
32
64
128
```

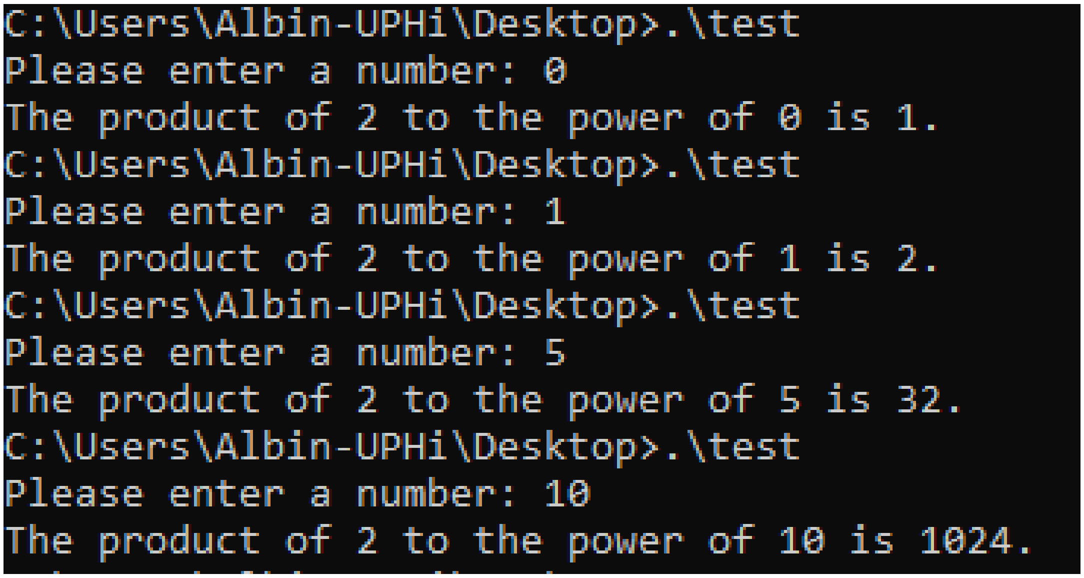
4. Write a code that computes for the power of two.

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



```c
#include <stdio.h>

int main()
{
    int i, power, answer = 1;

    printf("Please enter a number: ");
    scanf("%d", &power);

    i = power;          // sets a new variable i with which is equivalent to power
    while (i != 0){
        answer *= 2;
        i --;
    }

    // sets a new value of i, so the value of power will not change and can be recalled in the print statement
    printf("The product of 2 to the power of %d is %d.", power, answer);

    return 0;
}
```



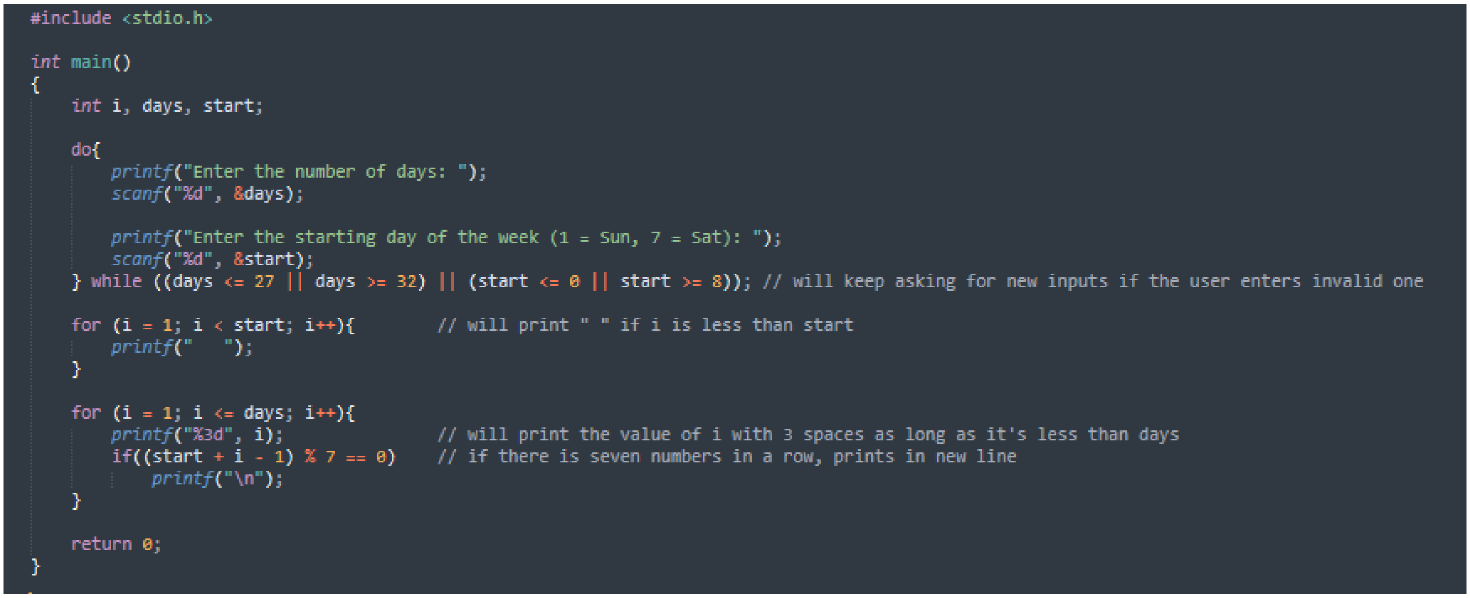
```
C:\Users\Albin-UPHi\Desktop>.\test
Please enter a number: 0
The product of 2 to the power of 0 is 1.
C:\Users\Albin-UPHi\Desktop>.\test
Please enter a number: 1
The product of 2 to the power of 1 is 2.
C:\Users\Albin-UPHi\Desktop>.\test
Please enter a number: 5
The product of 2 to the power of 5 is 32.
C:\Users\Albin-UPHi\Desktop>.\test
Please enter a number: 10
The product of 2 to the power of 10 is 1024.
```

5. Write a program that displays a one-month calendar.

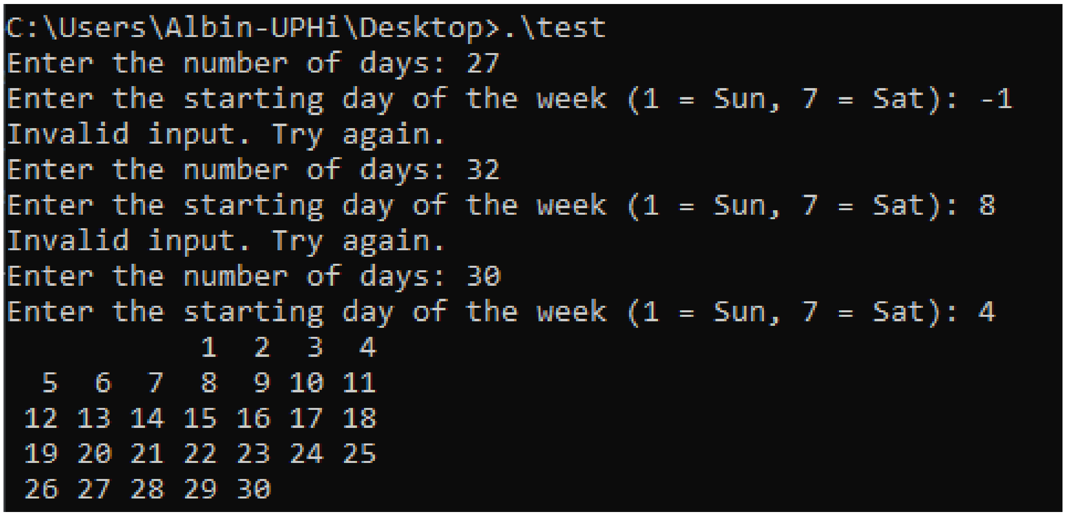
   There should be a user prompt to set:

   - 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.

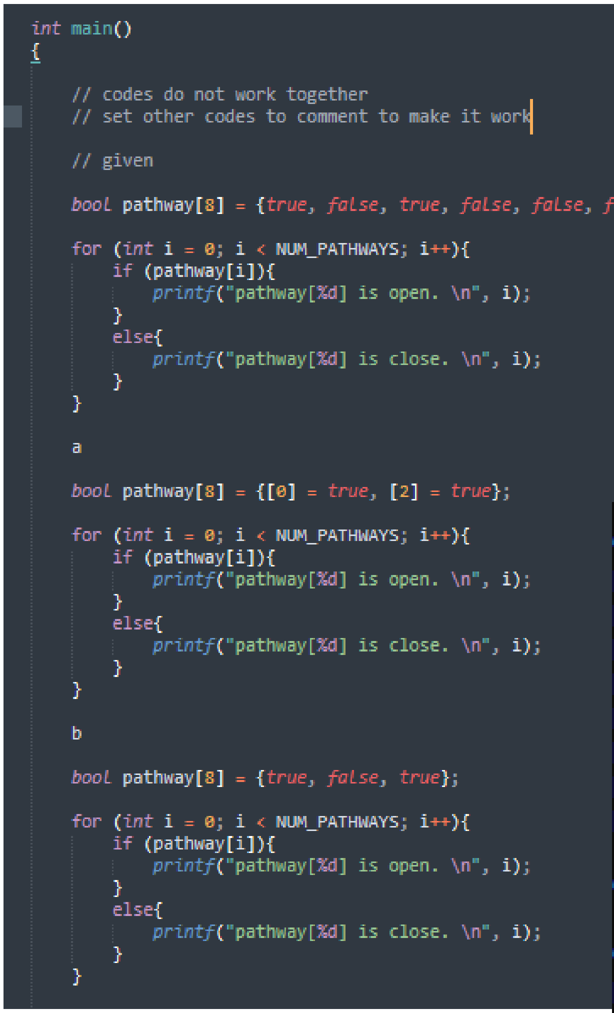


```c
#include <stdio.h>

int main()
{
    int i, days, start;

    do{
        printf("Enter the number of days: ");
        scanf("%d", &days);

        printf("Enter the starting day of the week (1 = Sun, 7 = Sat): ");
        scanf("%d", &start);
    } while ((days <= 27 || days >= 32) || (start <= 0 || start >= 8)); // will keep asking for new inputs if the user enters invalid one

    for (i = 1; i < start; i++){       // will print " " if i is less than start
        printf("   ");
    }

    for (i = 1; i <= days; i++){
        printf("%3d", i);               // will print the value of i with 3 spaces as long as it's less than days
        if((start + i - 1) % 7 == 0)    // if there is seven numbers in a row, prints in new line
            printf("\n");
    }

    return 0;
}
```



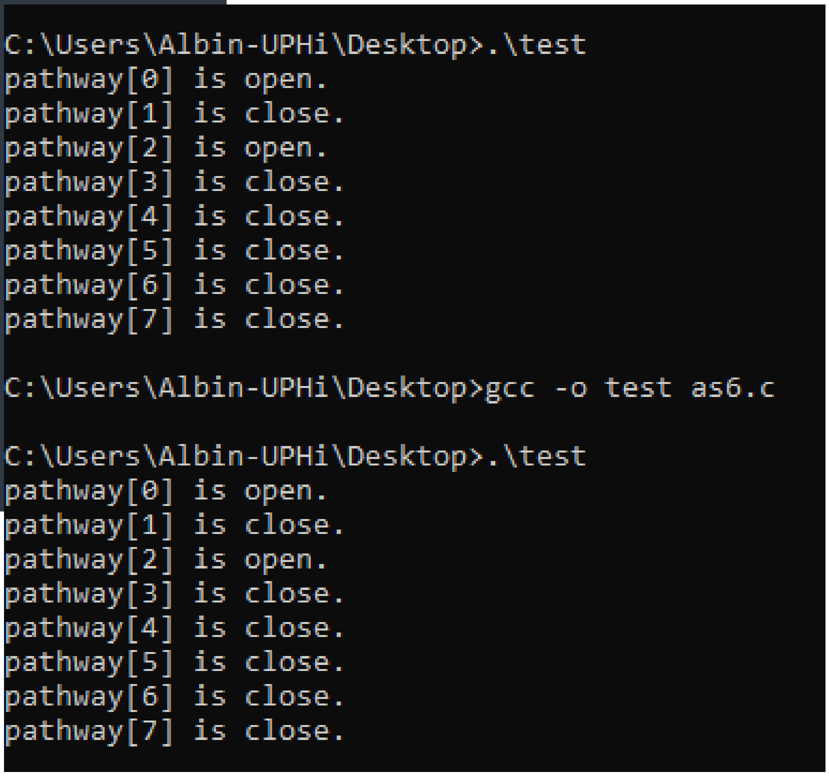
```
C:\Users\Albin-UPHi\Desktop>.\test
Enter the number of days: 27
Enter the starting day of the week (1 = Sun, 7 = Sat): -1
Invalid input. Try again.
Enter the number of days: 32
Enter the starting day of the week (1 = Sun, 7 = Sat): 8
Invalid input. Try again.
Enter the number of days: 30
Enter the starting day of the week (1 = Sun, 7 = Sat): 4
           1  2  3  4
  5  6  7  8  9 10 11
 12 13 14 15 16 17 18
 19 20 21 22 23 24 25
 26 27 28 29 30
```

6. In the program below, an array named pathway contains eight bool values. Each bool element refers to whether a pathway is open or close for transportation.
   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.
   b. Revise line 16 such that the initializer will be short as possible (without using a designated initializer).



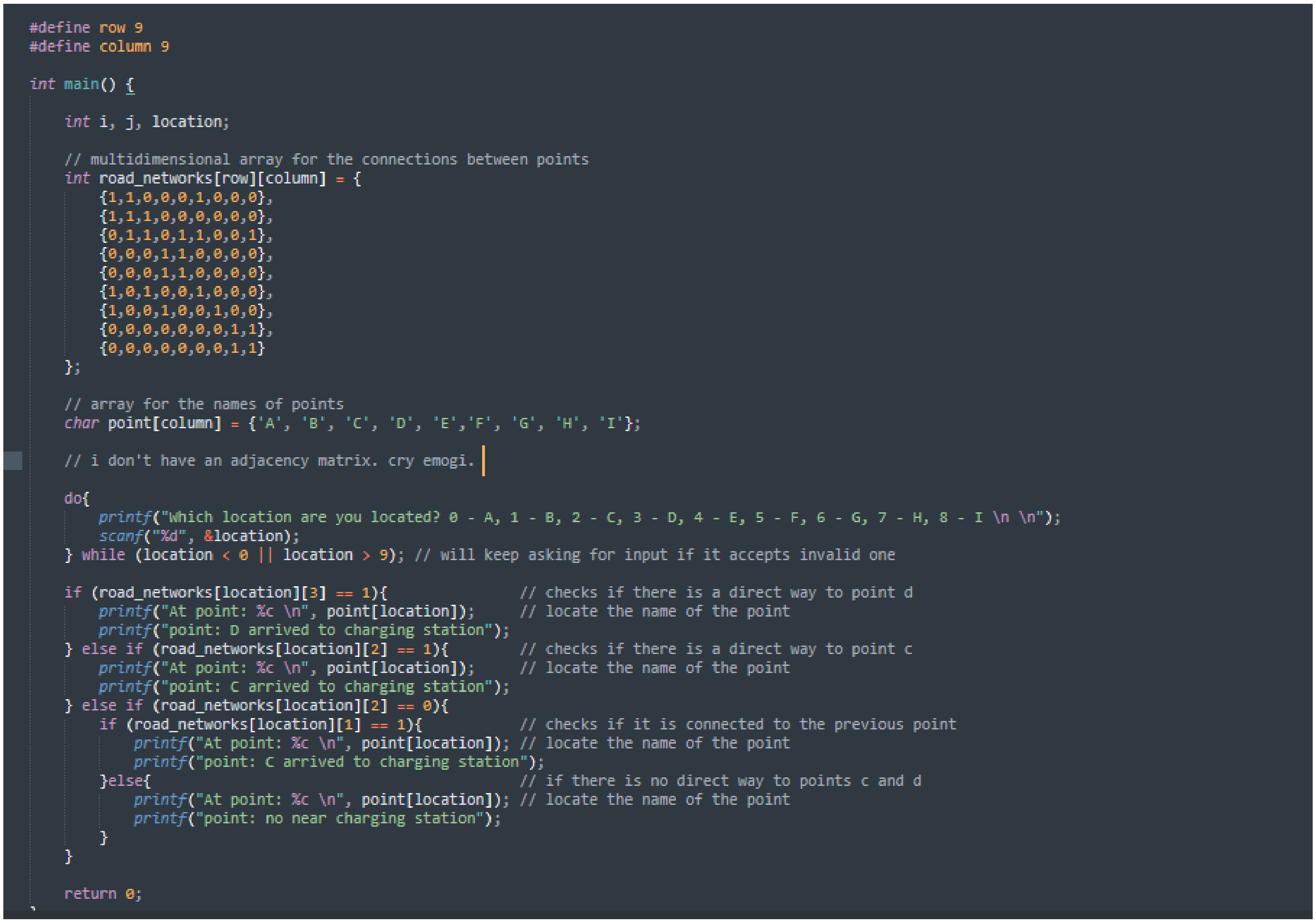
```
int main()
{

    // codes do not work together
    // set other codes to comment to make it work

    // given

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

    a

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

    b

    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);
        }
    }
```

Top: output for a, Bottom: output for b

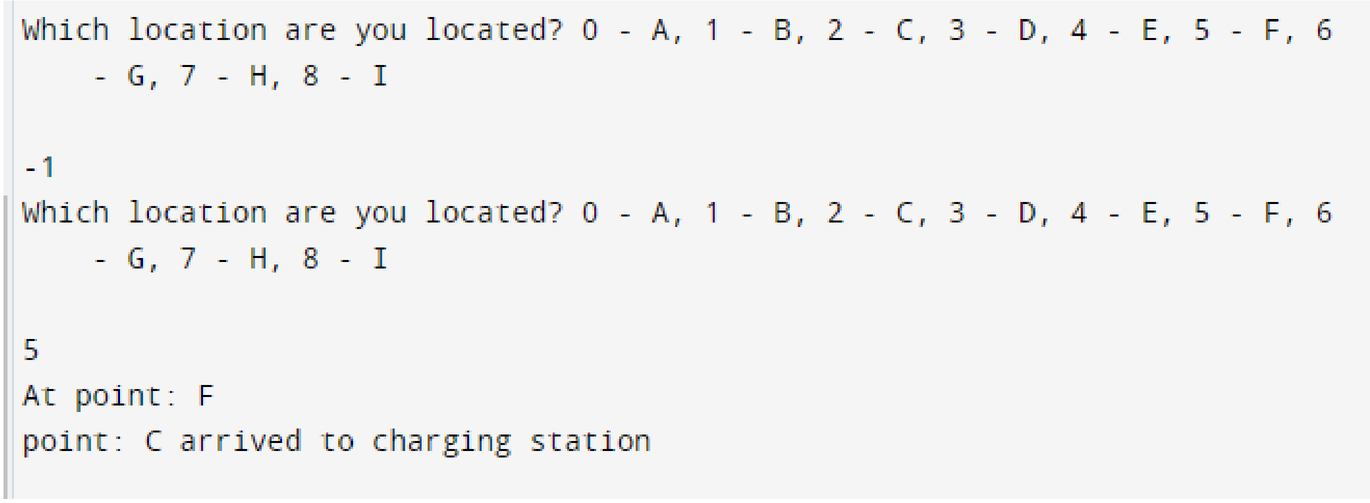


```
C:\Users\Albin-UPHi\Desktop>.\test
pathway[0] is open.
pathway[1] is close.
pathway[2] is open.
pathway[3] is close.
pathway[4] is close.
pathway[5] is close.
pathway[6] is close.
pathway[7] is close.

C:\Users\Albin-UPHi\Desktop>gcc -o test as6.c

C:\Users\Albin-UPHi\Desktop>.\test
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.
```

7. A road network can be represented using graphs. Assuming we have points / stations a, b, c, d, e, f, g, and h, we can represent a direct path from a point to another point using arrows.



```c
#define row 9
#define column 9

int main() {

    int i, j, location;

    // multidimensional array for the connections between points
    int road_networks[row][column] = {
        {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},
        {0,0,0,0,0,0,0,1,1},
        {0,0,0,0,0,0,0,1,1}
    };

    // array for the names of points
    char point[column] = {'A', 'B', 'C', 'D', 'E','F', 'G', 'H', 'I'};

    // i don't have an adjacency matrix. cry emogi.

    do{
        printf("Which location are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H, 8 - I \n \n");
        scanf("%d", &location);
    } while (location < 0 || location > 9); // will keep asking for input if it accepts invalid one

    if (road_networks[location][3] == 1){              // checks if there is a direct way to point d
        printf("At point: %c \n", point[location]);     // locate the name of the point
        printf("point: D arrived to charging station");
    } else if (road_networks[location][2] == 1){        // checks if there is a direct way to point c
        printf("At point: %c \n", point[location]);     // locate the name of the point
        printf("point: C arrived to charging station");
    } else if (road_networks[location][2] == 0){
        if (road_networks[location][1] == 1){           // checks if it is connected to the previous point
            printf("At point: %c \n", point[location]); // locate the name of the point
            printf("point: C arrived to charging station");
        }else{                                          // if there is no direct way to points c and d
            printf("At point: %c \n", point[location]); // locate the name of the point
            printf("point: no near charging station");
        }
    }

    return 0;
```



```
Which location are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6
    - G, 7 - H, 8 - I


-1
Which location are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6
    - G, 7 - H, 8 - I


5
At point: F
point: C arrived to charging station
```