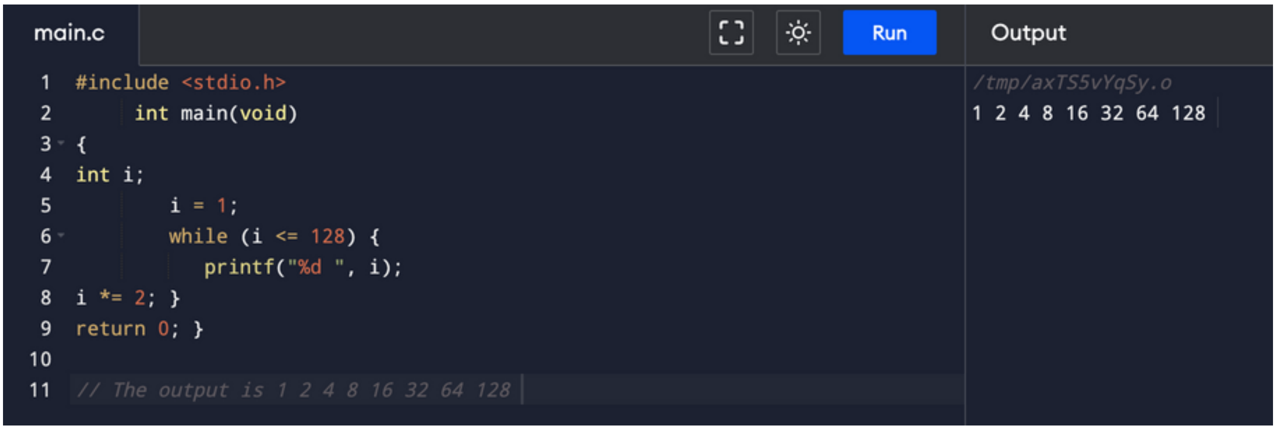
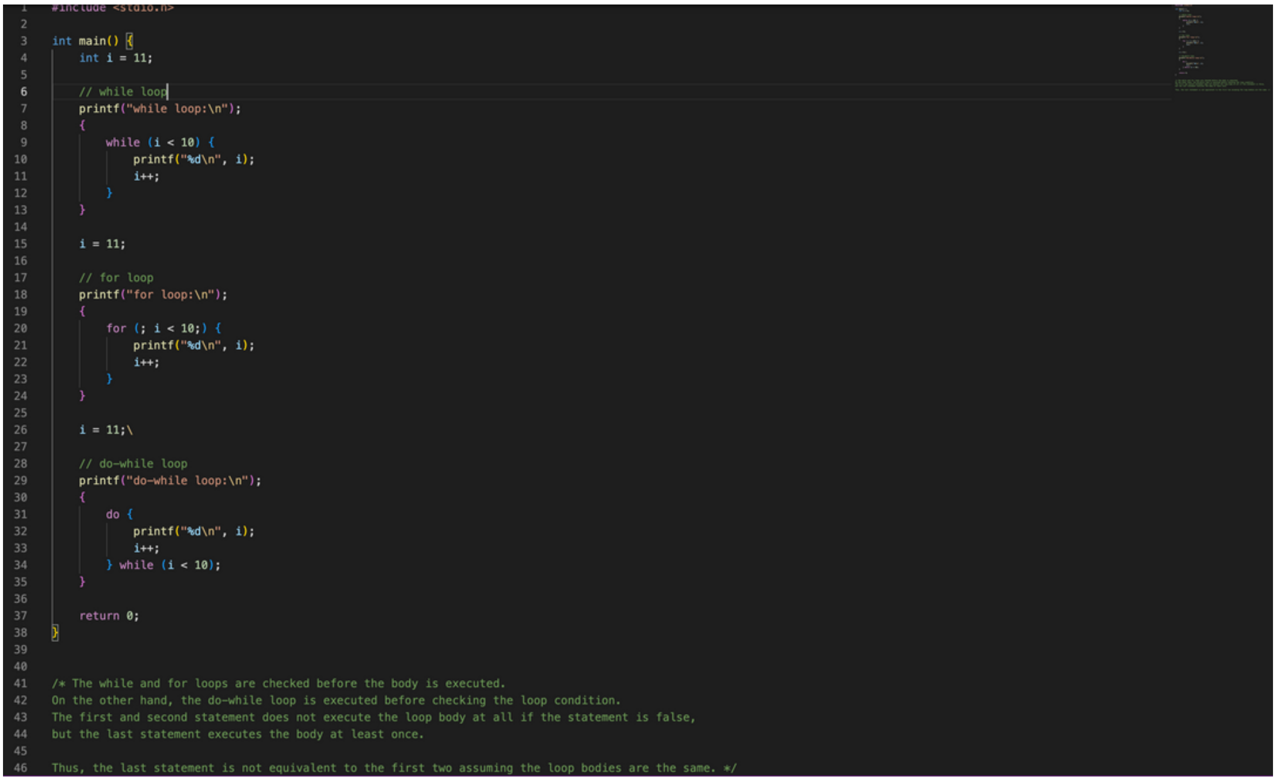
1. code and output



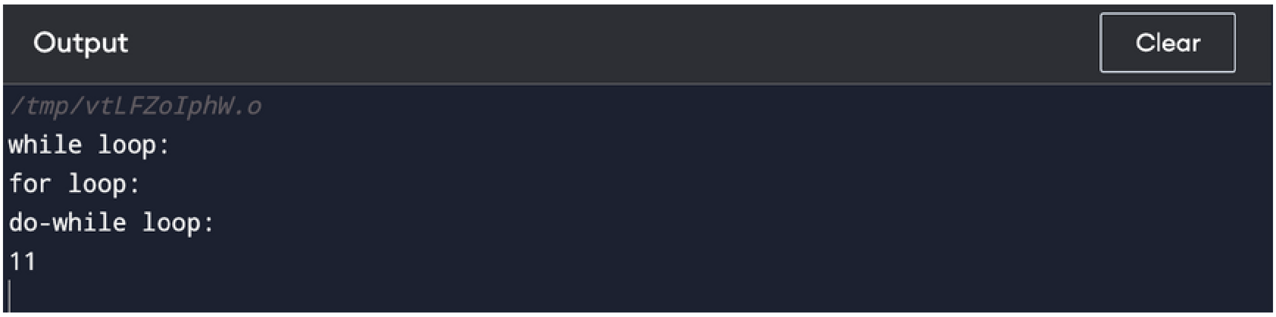
```c
#include <stdio.h>
      int main(void)
{
int i;
          i = 1;
          while (i <= 128) {
             printf("%d ", i);
i *= 2; }
return 0; }

// The output is 1 2 4 8 16 32 64 128
```

```
/tmp/axTS5vYqSy.o
1 2 4 8 16 32 64 128
```

2. code and output

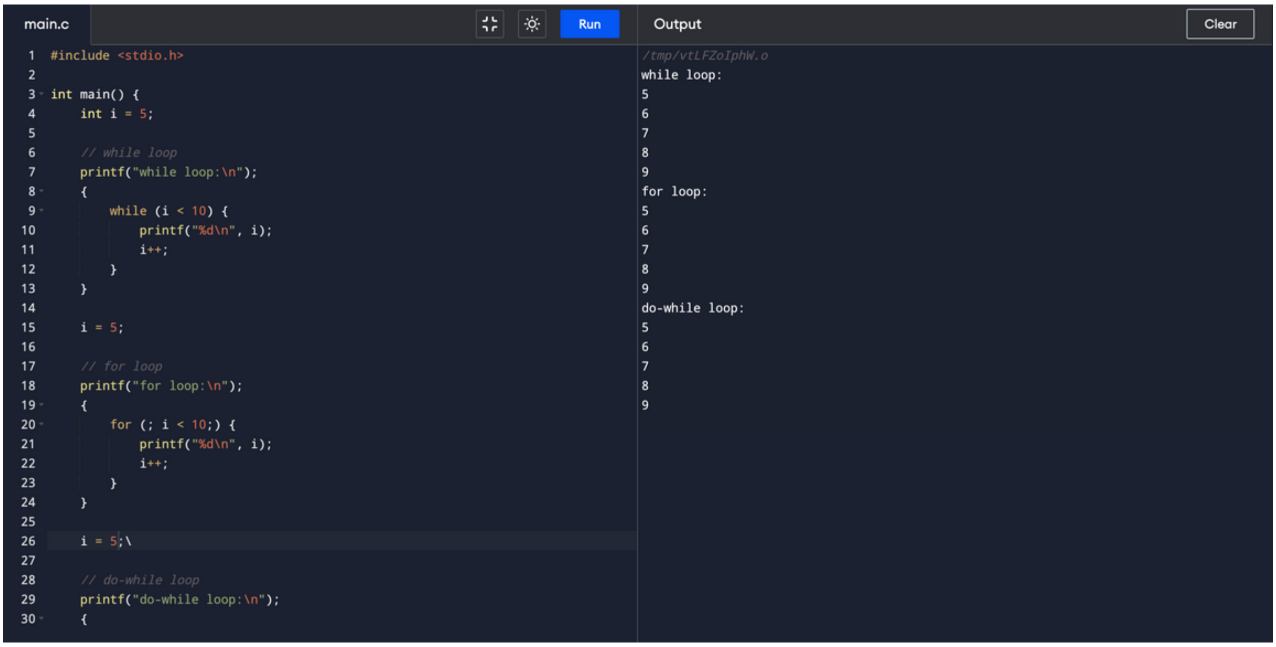


```c
#include <stdio.h>

int main() {
    int i = 11;

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

    i = 11;

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

    i = 11;\

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

    return 0;
}


/* The while and for loops are checked before the body is executed.
On the other hand, the do-while loop is executed before checking the loop condition.
The first and second statement does not execute the loop body at all if the statement is false,
but the last statement executes the body at least once.

Thus, the last statement is not equivalent to the first two assuming the loop bodies are the same. */
```



```
/tmp/vtLFZoIphW.o
while loop:
for loop:
do-while loop:
11
```

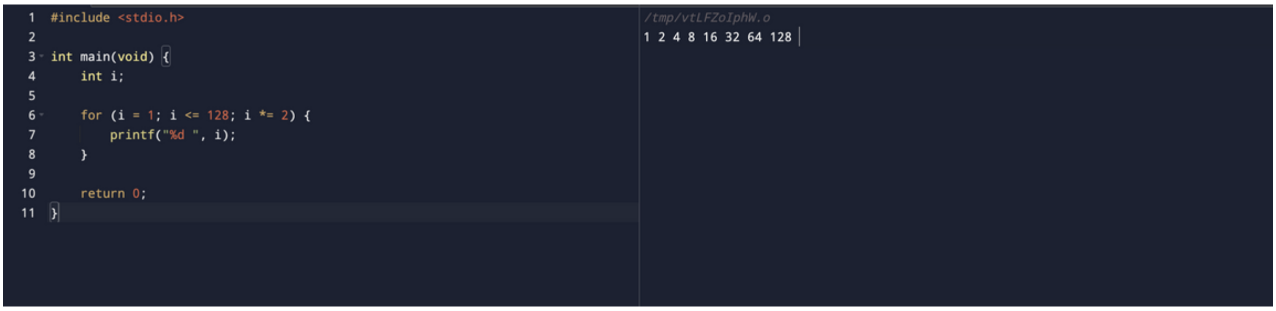
other output example



```c
#include <stdio.h>

int main() {
    int i = 5;

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

    i = 5;

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

    i = 5;\

    // do-while loop
    printf("do-while loop:\n");
    {
```

```
/tmp/vtLFZoIphW.o
while loop:
5
6
7
8
9
for loop:
5
6
7
8
9
do-while loop:
5
6
7
8
9
```

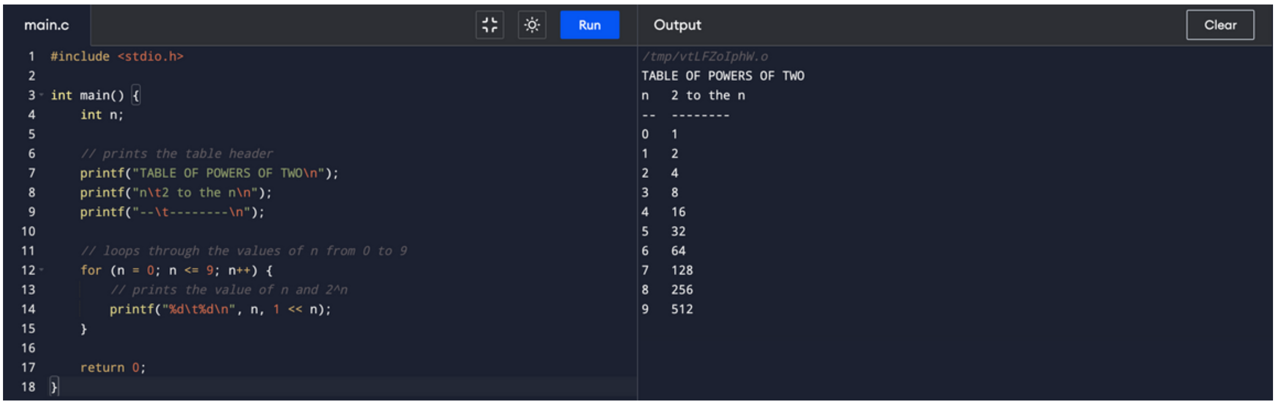
3. code and output



```c
#include <stdio.h>

int main(void) {
    int i;

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

    return 0;
}
```

```
/tmp/vtLFZoIphW.o
1 2 4 8 16 32 64 128
```

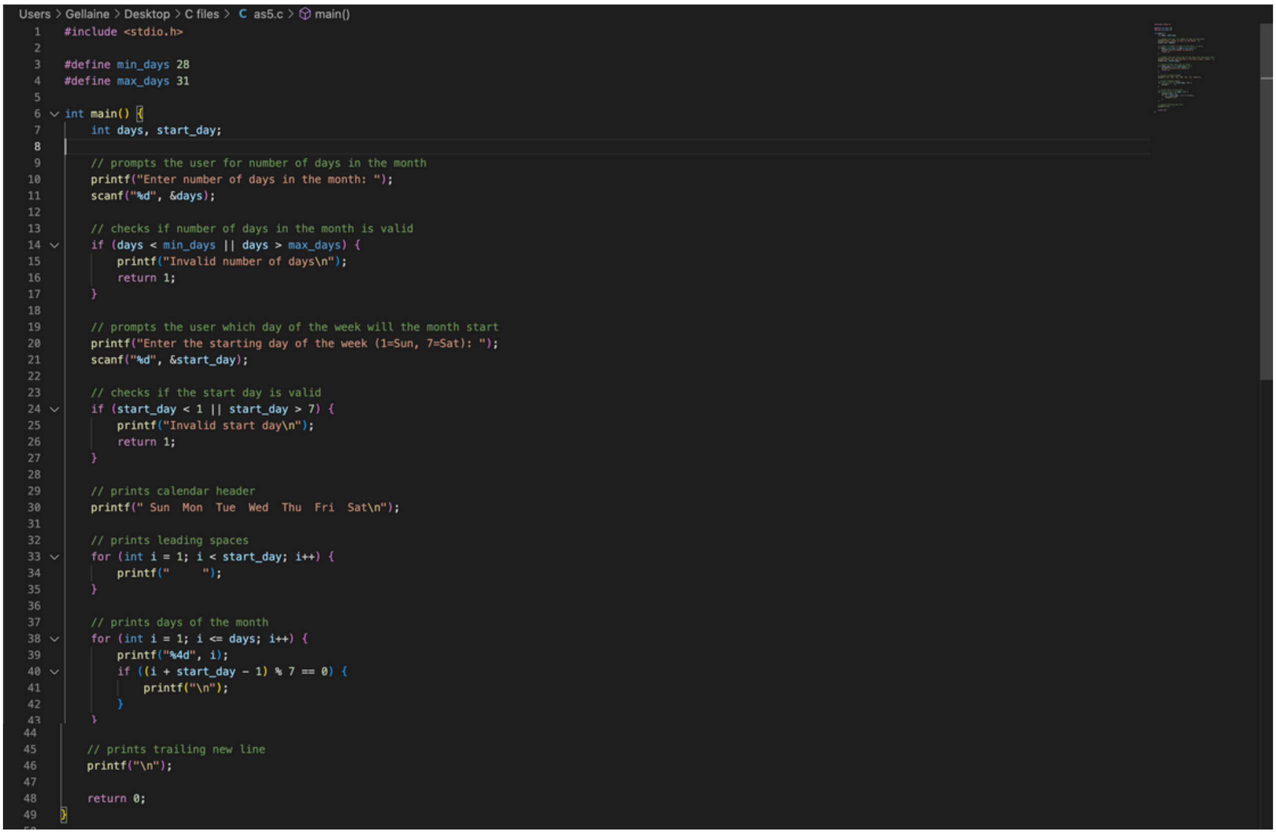
4. code and output



```c
#include <stdio.h>

int main() {
    int n;

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

    // loops through the values of n from 0 to 9
    for (n = 0; n <= 9; n++) {
        // prints the value of n and 2^n
        printf("%d\t%d\n", n, 1 << n);
    }

    return 0;
}
```

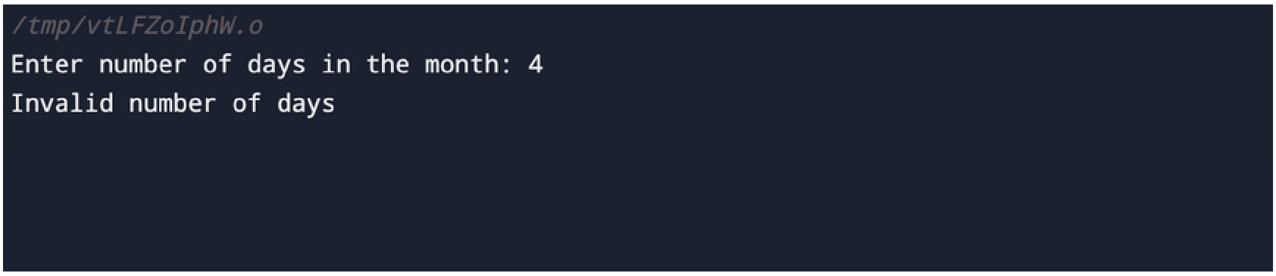
```
/tmp/vtLFZoIphW.o
TABLE OF POWERS OF TWO
n   2 to the n
--  --------
0   1
1   2
2   4
3   8
4   16
5   32
6   64
7   128
8   256
9   512
```

5. code and output

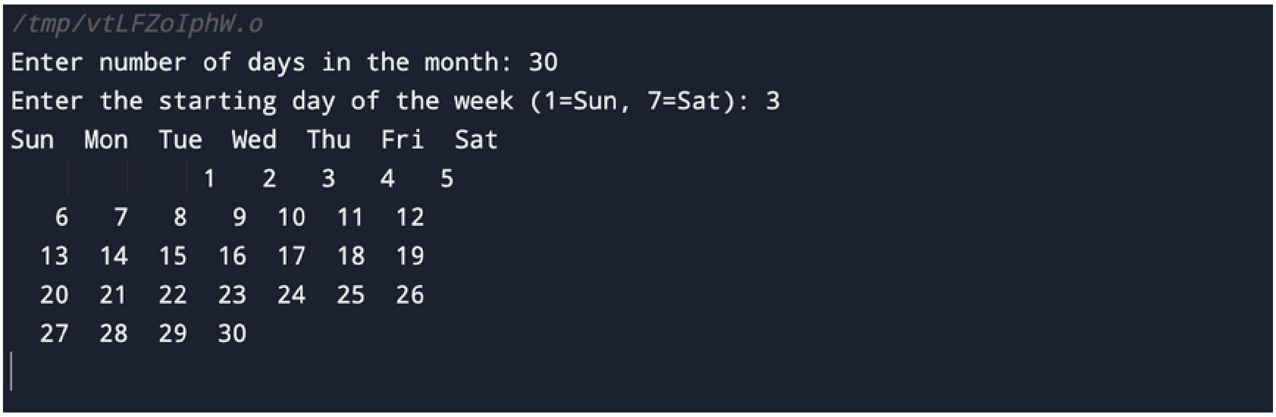


Users > Gellaine > Desktop > C files > as5.c > main()

```c
#include <stdio.h>

#define min_days 28
#define max_days 31

int main() {
    int days, start_day;

    // prompts the user for number of days in the month
    printf("Enter number of days in the month: ");
    scanf("%d", &days);

    // checks if number of days in the month is valid
    if (days < min_days || days > max_days) {
        printf("Invalid number of days\n");
        return 1;
    }

    // prompts the user which day of the week will the month start
    printf("Enter the starting day of the week (1=Sun, 7=Sat): ");
    scanf("%d", &start_day);

    // checks if the start day is valid
    if (start_day < 1 || start_day > 7) {
        printf("Invalid start day\n");
        return 1;
    }

    // prints calendar header
    printf(" Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

    // prints leading spaces
    for (int i = 1; i < start_day; i++) {
        printf("    ");
    }

    // prints days of the month
    for (int i = 1; i <= days; i++) {
        printf("%4d", i);
        if ((i + start_day - 1) % 7 == 0) {
            printf("\n");
        }
    }

    // prints trailing new line
    printf("\n");

    return 0;
}
```

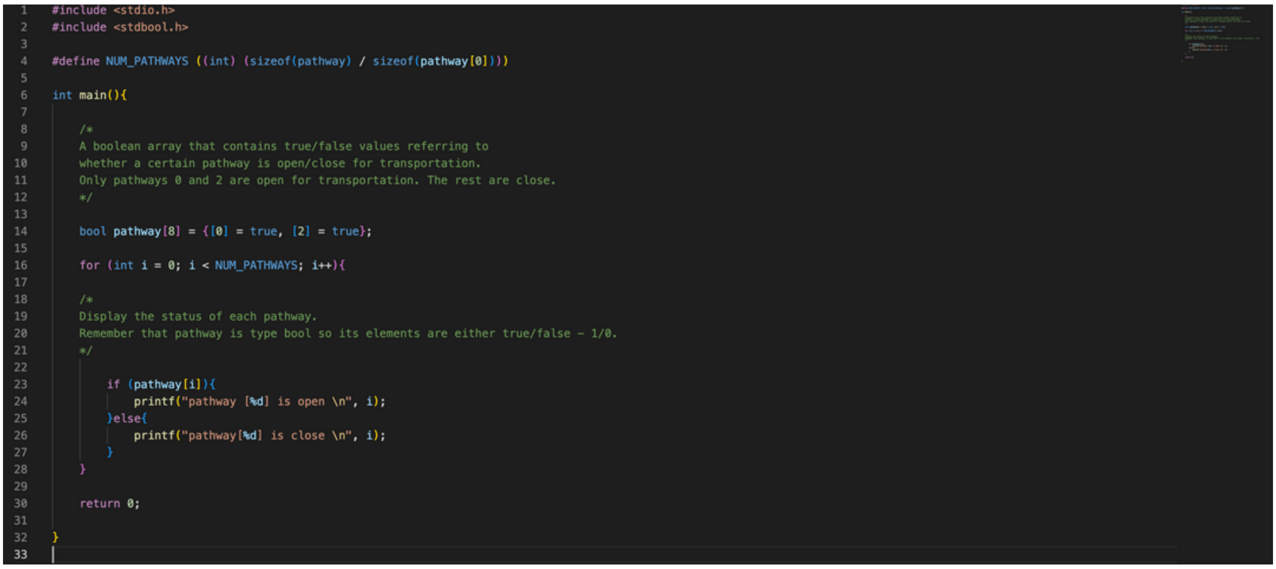


```
/tmp/vtLFZoIphW.o
Enter number of days in the month: 4
Invalid number of days
```

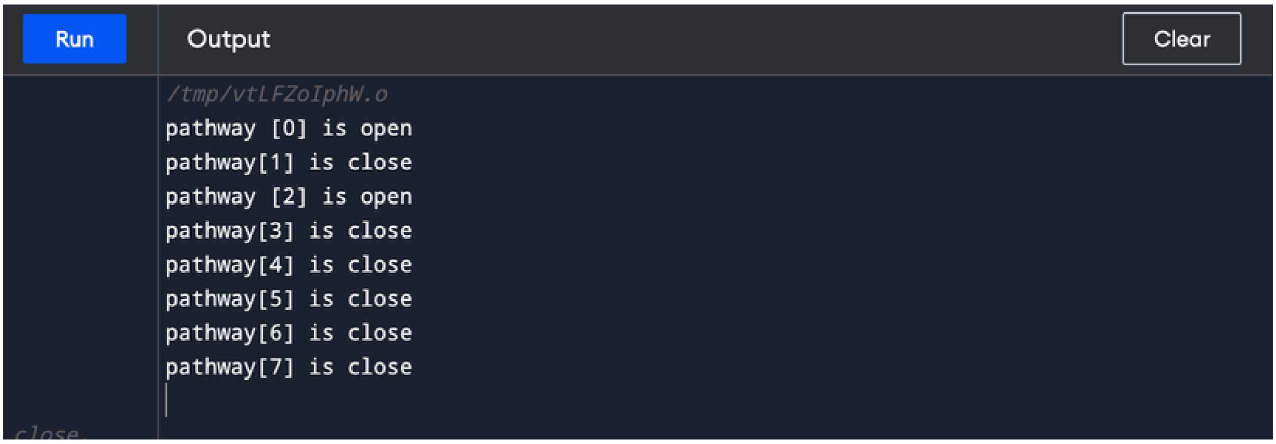


```
/tmp/vtLFZoIphW.o
Enter number of days in the month: 30
Enter the starting day of the week (1=Sun, 7=Sat): 3
Sun  Mon  Tue  Wed  Thu  Fri  Sat
             1   2   3   4   5
   6   7   8   9  10  11  12
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29  30
```

6a. code and output

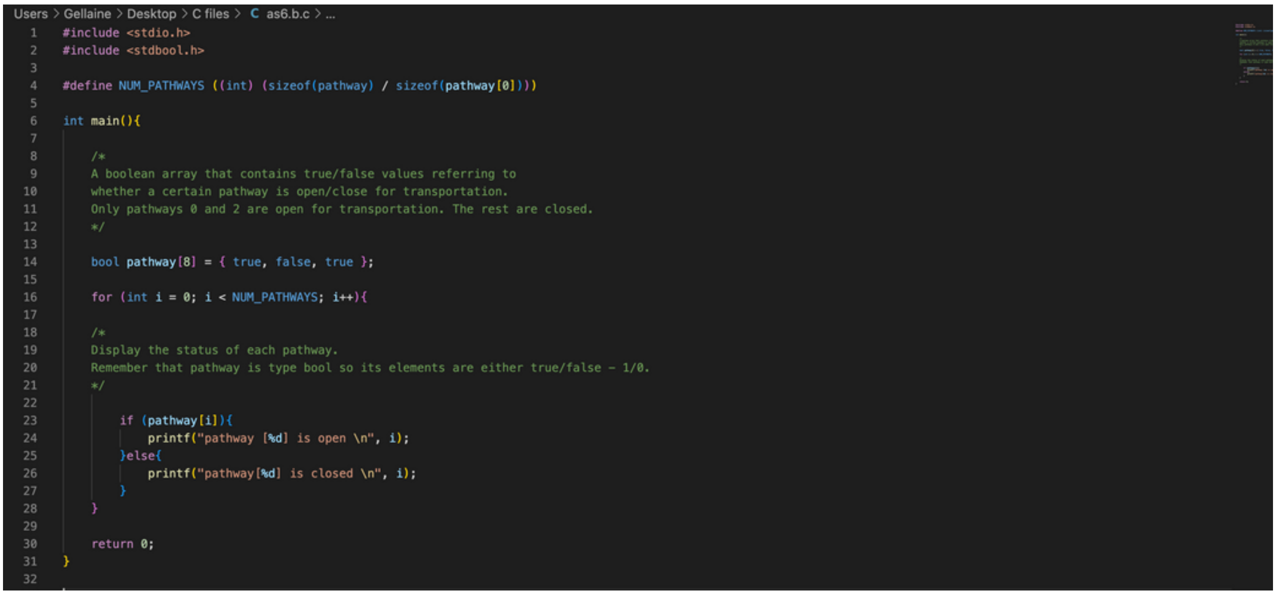


```c
#include <stdio.h>
#include <stdbool.h>

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

int main(){

    /*
    A boolean array that contains true/false values referring to
    whether a certain pathway is open/close for transportation.
    Only pathways 0 and 2 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 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;

}
```

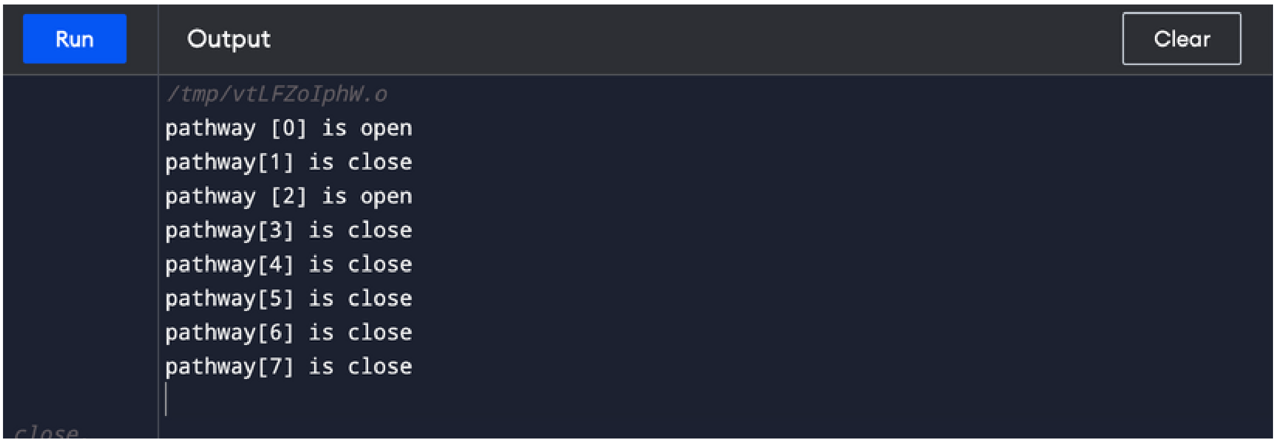


```
/tmp/vtLFZoIphW.o
pathway [0] is open
pathway[1] is close
pathway [2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

6b. code and output

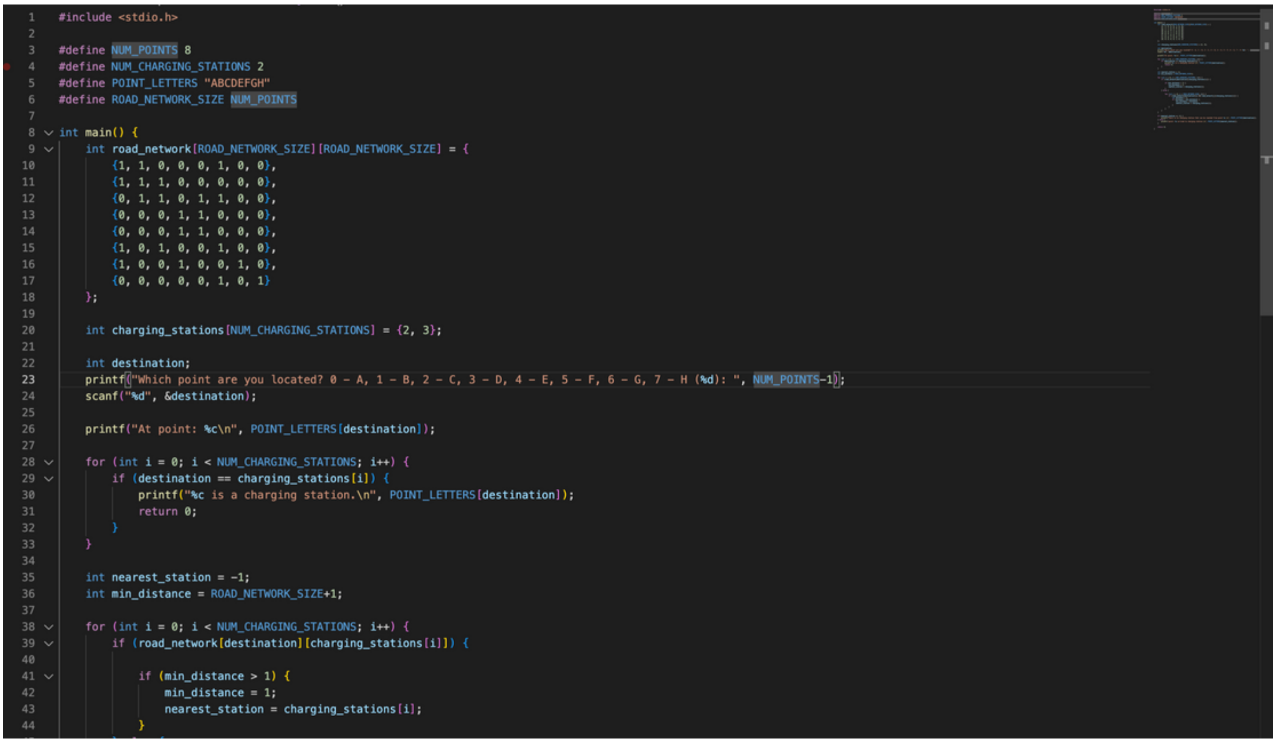


```c
#include <stdio.h>
#include <stdbool.h>

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

int main(){

    /*
    A boolean array that contains true/false values 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 };

    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;
}
```

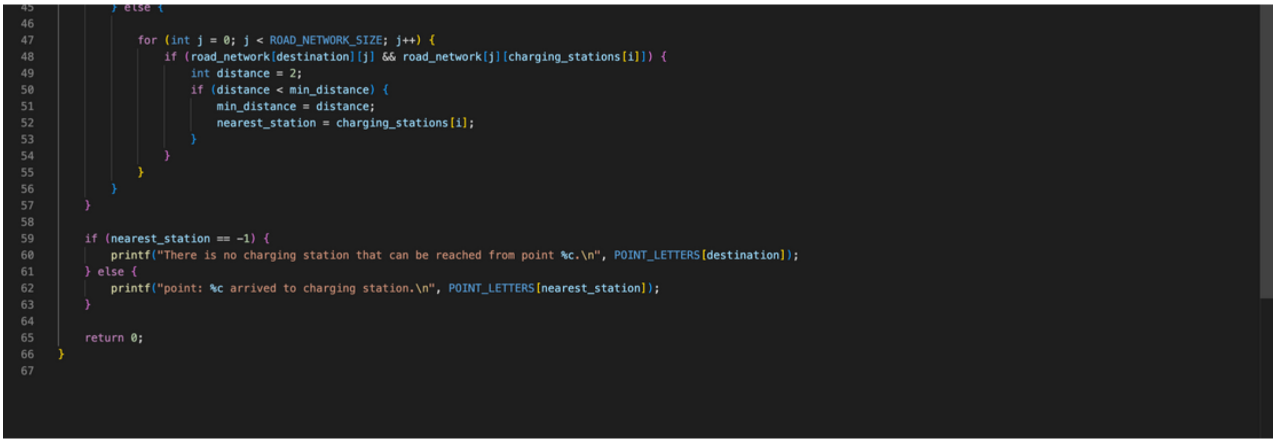


```
/tmp/vtLFZoIphW.o
pathway [0] is open
pathway[1] is close
pathway [2] is open
pathway[3] is close
pathway[4] is close
pathway[5] is close
pathway[6] is close
pathway[7] is close
```

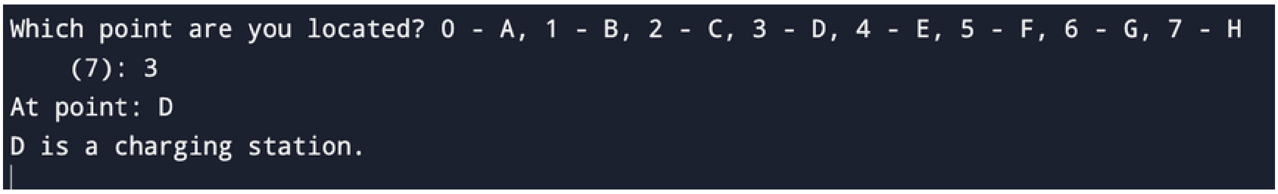
7. code and output



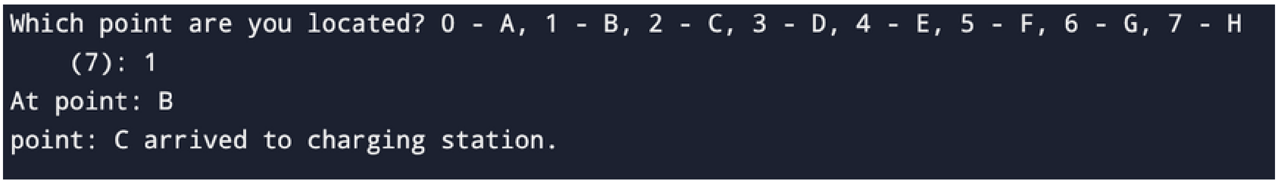
```c
#include <stdio.h>

#define NUM_POINTS 8
#define NUM_CHARGING_STATIONS 2
#define POINT_LETTERS "ABCDEFGH"
#define ROAD_NETWORK_SIZE NUM_POINTS

int main() {
    int road_network[ROAD_NETWORK_SIZE][ROAD_NETWORK_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}
    };

    int charging_stations[NUM_CHARGING_STATIONS] = {2, 3};

    int destination;
    printf("Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H (%d): ", NUM_POINTS-1);
    scanf("%d", &destination);

    printf("At point: %c\n", POINT_LETTERS[destination]);

    for (int i = 0; i < NUM_CHARGING_STATIONS; i++) {
        if (destination == charging_stations[i]) {
            printf("%c is a charging station.\n", POINT_LETTERS[destination]);
            return 0;
        }
    }

    int nearest_station = -1;
    int min_distance = ROAD_NETWORK_SIZE+1;

    for (int i = 0; i < NUM_CHARGING_STATIONS; i++) {
        if (road_network[destination][charging_stations[i]]) {

            if (min_distance > 1) {
                min_distance = 1;
                nearest_station = charging_stations[i];
            }
```



```c
        } else {

            for (int j = 0; j < ROAD_NETWORK_SIZE; j++) {
                if (road_network[destination][j] && road_network[j][charging_stations[i]]) {
                    int distance = 2;
                    if (distance < min_distance) {
                        min_distance = distance;
                        nearest_station = charging_stations[i];
                    }
                }
            }
        }
    }

    if (nearest_station == -1) {
        printf("There is no charging station that can be reached from point %c.\n", POINT_LETTERS[destination]);
    } else {
        printf("point: %c arrived to charging station.\n", POINT_LETTERS[nearest_station]);
    }

    return 0;
}
```



```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
    (7): 3
At point: D
D is a charging station.
```



```
Which point are you located? 0 - A, 1 - B, 2 - C, 3 - D, 4 - E, 5 - F, 6 - G, 7 - H
    (7): 1
At point: B
point: C arrived to charging station.
```