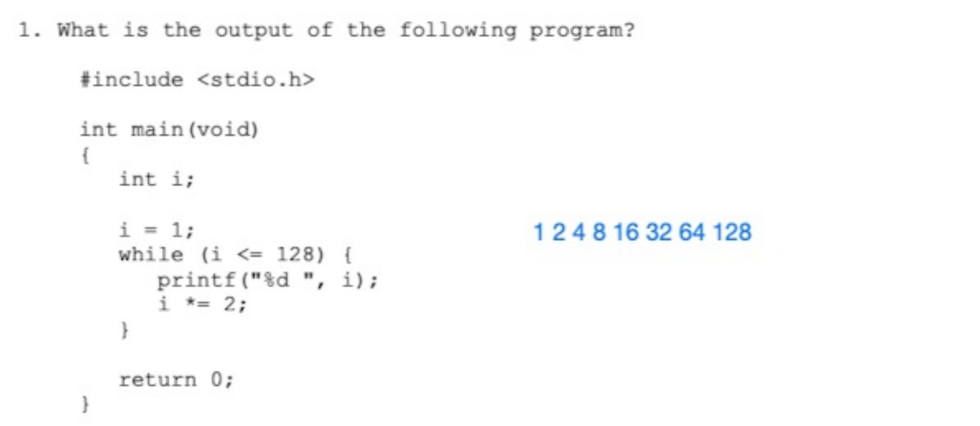
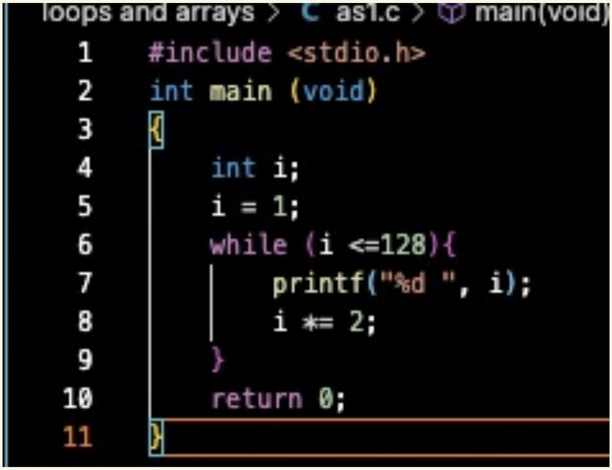


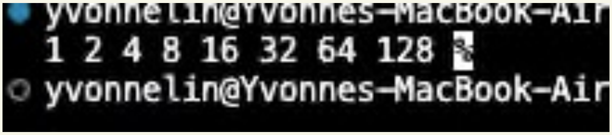
1. What is the output of the following program?

```
#include <stdio.h>

int main(void)
{
   int i;

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

   return 0;
}
```

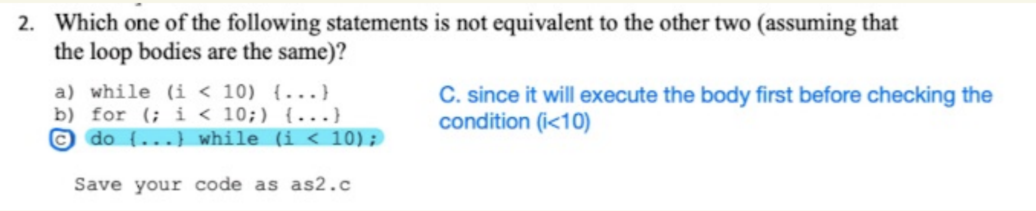
1 2 4 8 16 32 64 128



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



```
yvonnelin@Yvonnes-MacBook-Air
1 2 4 8 16 32 64 128 %
yvonnelin@Yvonnes-MacBook-Air
```



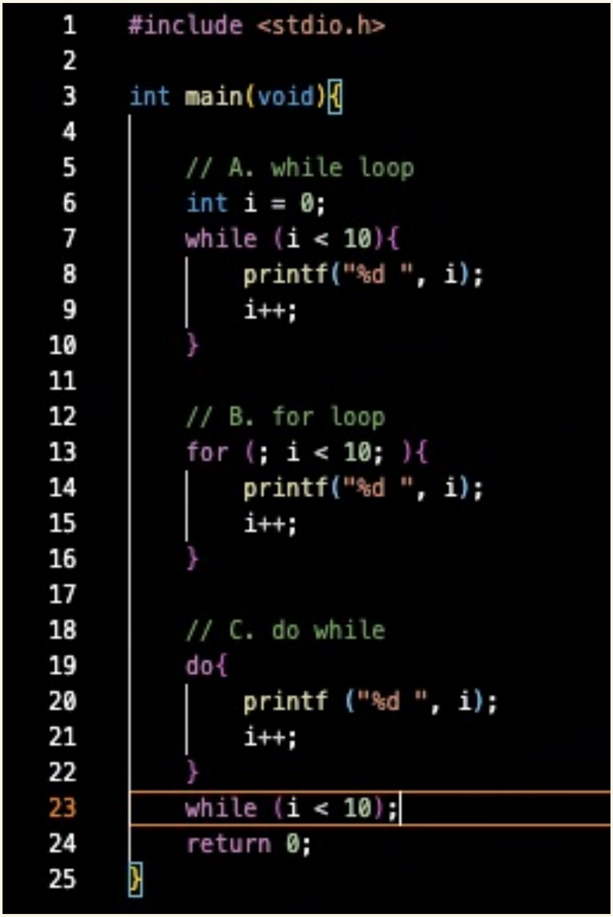
2. Which one of the following statements is not equivalent to the other two (assuming that the loop bodies are the same)?

```
a) while (i < 10) {...}
b) for (; i < 10;) {...}
```

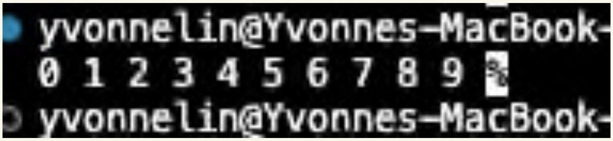
[✓ c) `do {...} while (i < 10);`]

C. since it will execute the body first before checking the condition (i<10)

```
Save your code as as2.c
```



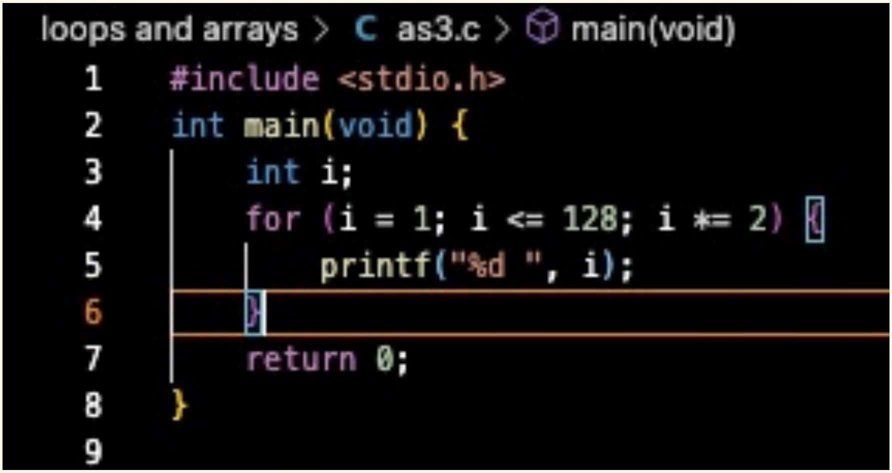
```c
#include <stdio.h>

int main(void){

    // A. while loop
    int i = 0;
    while (i < 10){
        printf("%d ", i);
        i++;
    }

    // B. for loop
    for (; i < 10; ){
        printf("%d ", i);
        i++;
    }

    // C. do while
    do{
        printf ("%d ", i);
        i++;
    }
    while (i < 10);
    return 0;
}
```



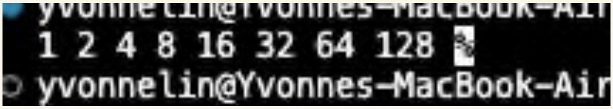
```
yvonnelin@Yvonnes-MacBook-
0 1 2 3 4 5 6 7 8 9 %
yvonnelin@Yvonnes-MacBook-
```

. 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.

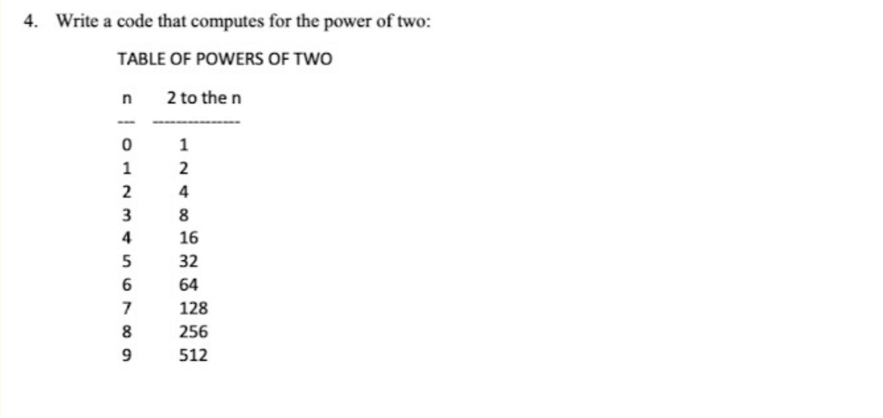
```
Save your code as as3.c
```



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

```

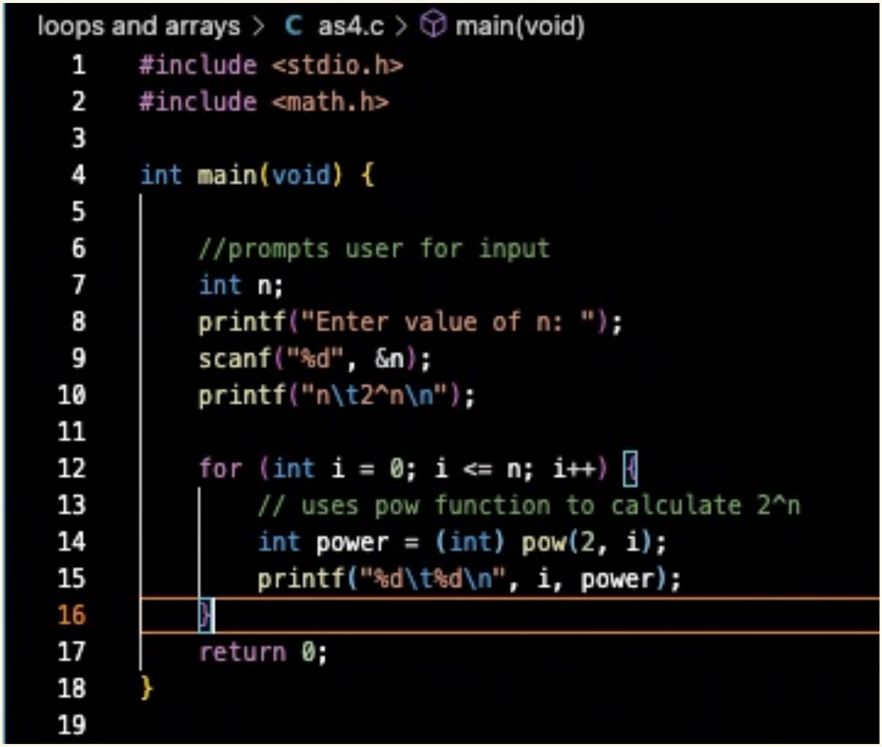


```
yvonnelin@Yvonnes-MacBook-Air
1 2 4 8 16 32 64 128 %
yvonnelin@Yvonnes-MacBook-Air
```



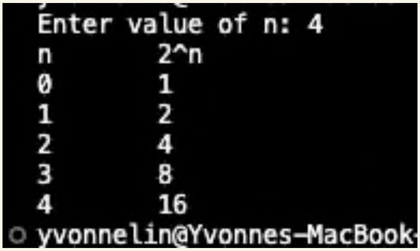
4. Write a code that computes for the power of two:

```
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
```

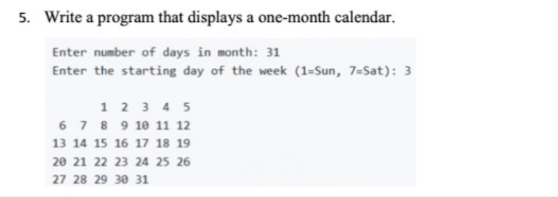


loops and arrays > C as4.c > main(void)

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

int main(void) {

    //prompts user for input
    int n;
    printf("Enter value of n: ");
    scanf("%d", &n);
    printf("n\t2^n\n");

    for (int i = 0; i <= n; i++) {
        // uses pow function to calculate 2^n
        int power = (int) pow(2, i);
        printf("%d\t%d\n", i, power);
    }
    return 0;
}

```

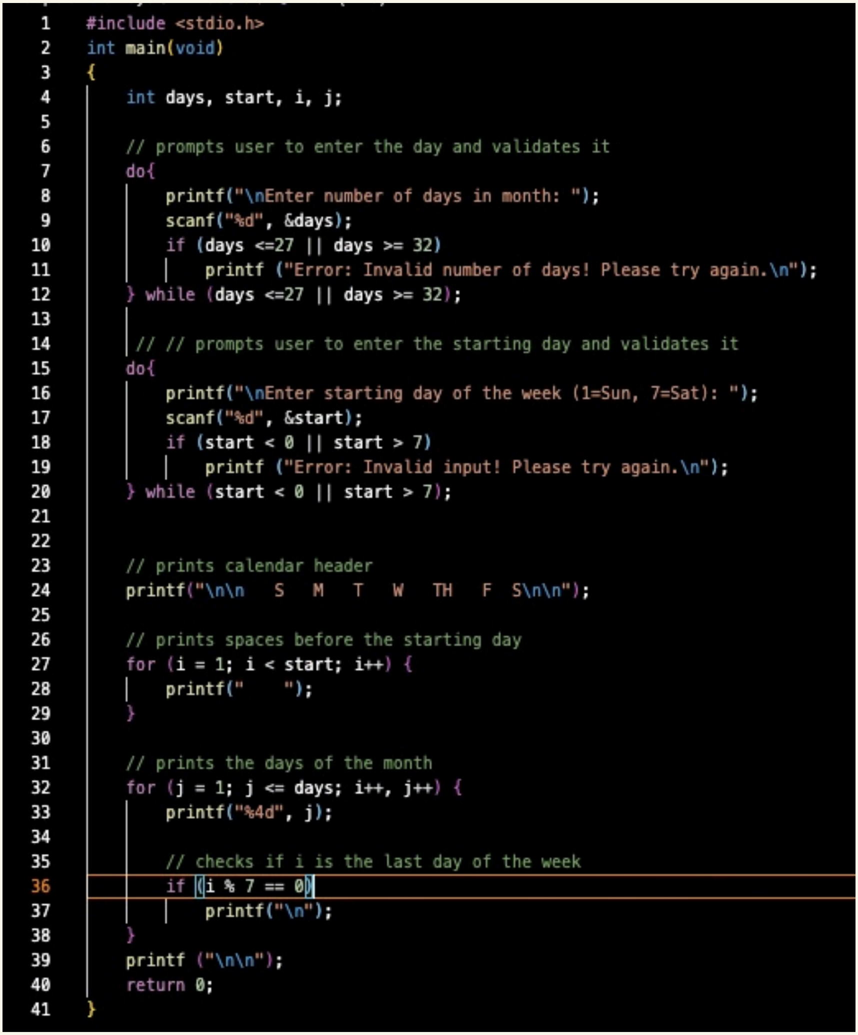


```
Enter value of n: 4
n       2^n
0       1
1       2
2       4
3       8
4       16
○ yvonnelin@Yvonnes-MacBook-
```

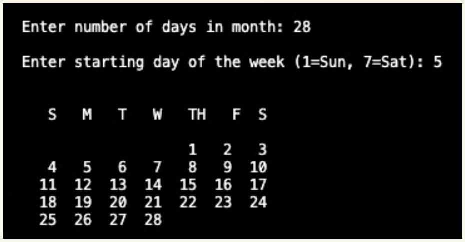


5. Write a program that displays a one-month calendar.

```
Enter number of days in month: 31
Enter the starting day of the week (1=Sun, 7=Sat): 3

       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 31
```



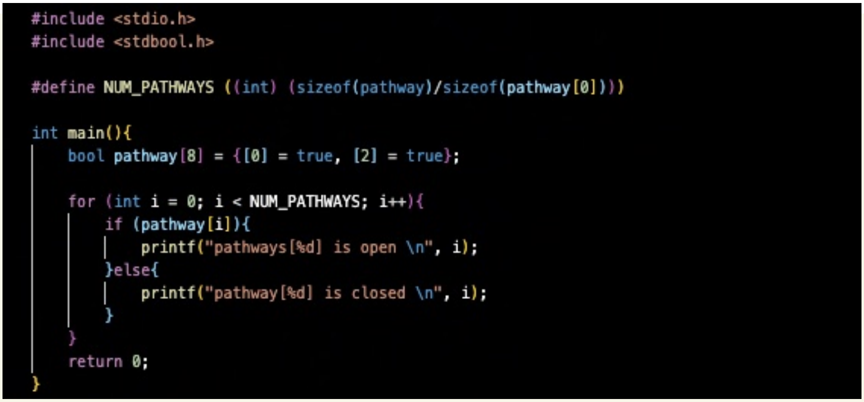
```c
#include <stdio.h>
int main(void)
{
    int days, start, i, j;

    // prompts user to enter the day and validates it
    do{
        printf("\nEnter number of days in month: ");
        scanf("%d", &days);
        if (days <=27 || days >= 32)
            printf ("Error: Invalid number of days! Please try again.\n");
    } while (days <=27 || days >= 32);

    // // prompts user to enter the starting day and validates it
    do{
        printf("\nEnter starting day of the week (1=Sun, 7=Sat): ");
        scanf("%d", &start);
        if (start < 0 || start > 7)
            printf ("Error: Invalid input! Please try again.\n");
    } while (start < 0 || start > 7);


    // prints calendar header
    printf("\n\n   S   M   T   W   TH   F  S\n\n");

    // prints spaces before the starting day
    for (i = 1; i < start; i++) {
        printf("    ");
    }

    // prints the days of the month
    for (j = 1; j <= days; i++, j++) {
        printf("%4d", j);

        // checks if i is the last day of the week
        if (i % 7 == 0)
            printf("\n");
    }
    printf ("\n\n");
    return 0;
}
```



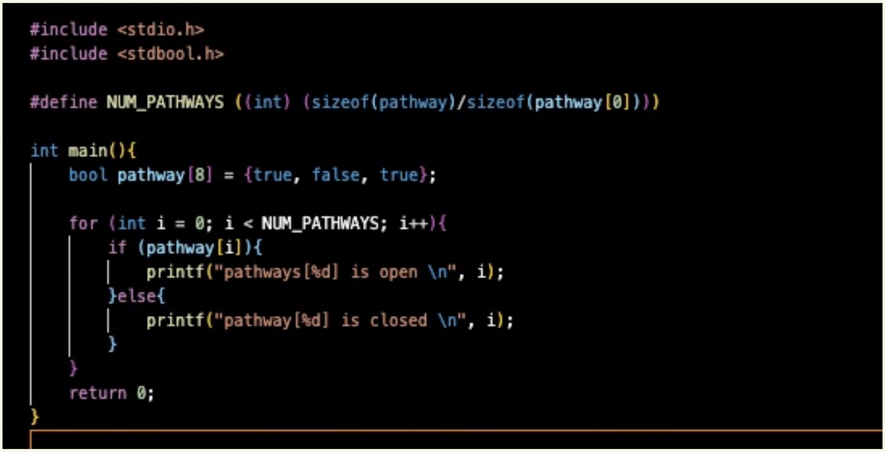
```
Enter number of days in month: 28

Enter starting day of the week (1=Sun, 7=Sat): 5


   S   M   T   W   TH   F  S

                   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
```

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)

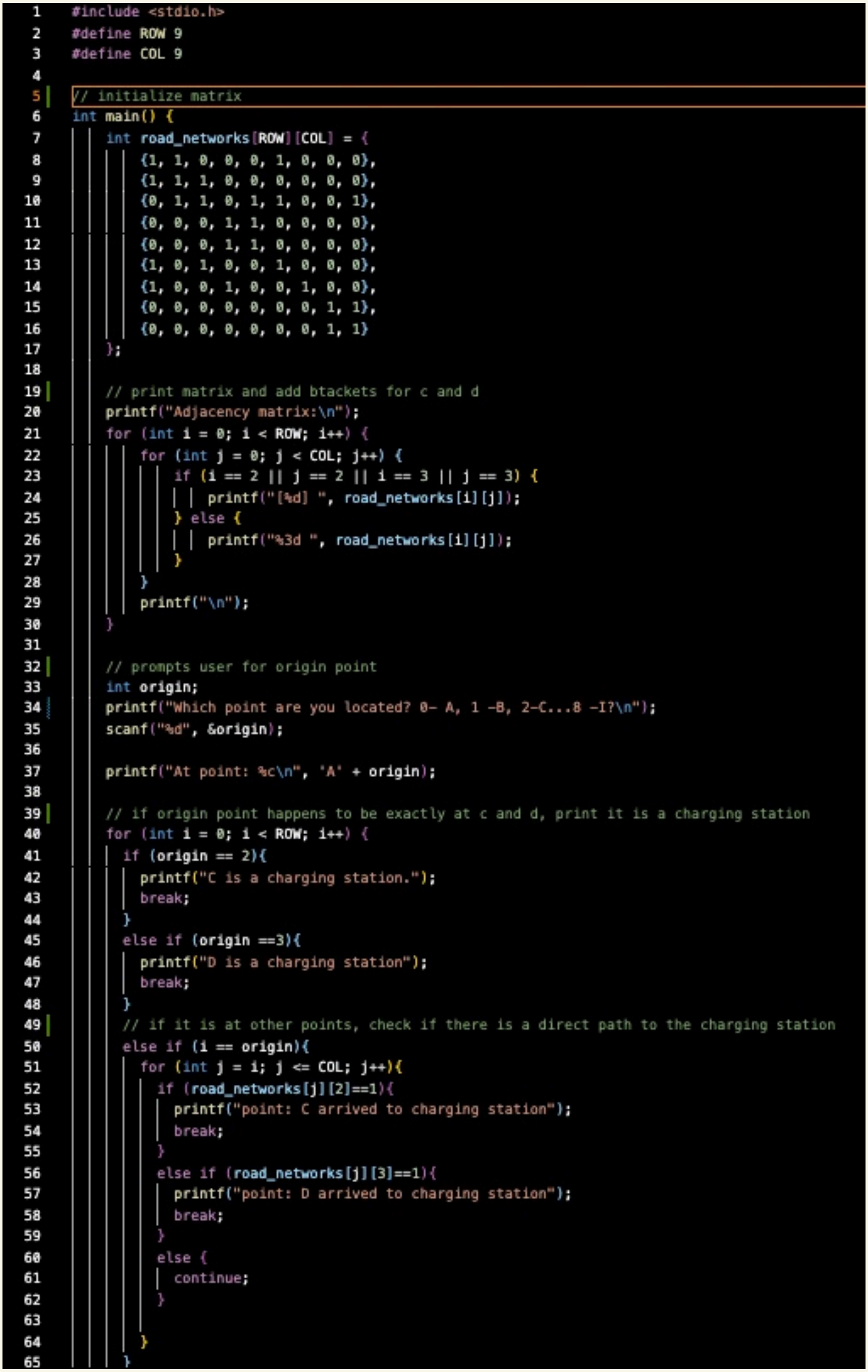


```
#include <stdio.h>
#include <stdbool.h>

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

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

    for (int i = 0; i < NUM_PATHWAYS; i++){
        if (pathway[i]){
            printf("pathways[%d] is open \n", i);
        }else{
            printf("pathway[%d] is closed \n", i);
        }
    }
    return 0;
}
```



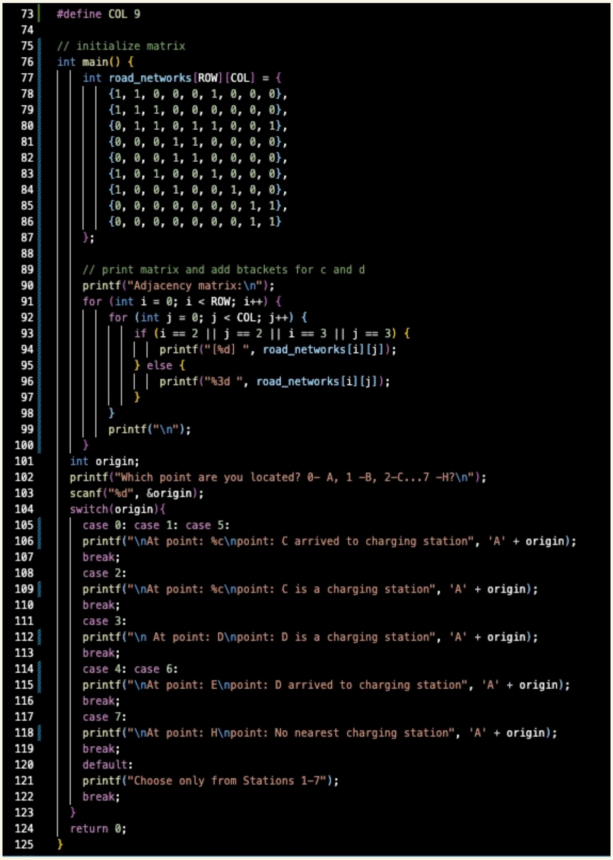
```
#include <stdio.h>
#include <stdbool.h>

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

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

    for (int i = 0; i < NUM_PATHWAYS; i++){
        if (pathway[i]){
            printf("pathways[%d] is open \n", i);
        }else{
            printf("pathway[%d] is closed \n", i);
        }
    }
    return 0;
}
```

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.



```
#include <stdio.h>
#define ROW 9
#define COL 9

// initialize matrix
int main() {
    int road_networks[ROW][COL] = {
        {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}
    };

    // print matrix and add btackets for c and d
    printf("Adjacency matrix:\n");
    for (int i = 0; i < ROW; i++) {
        for (int j = 0; j < COL; j++) {
            if (i == 2 || j == 2 || i == 3 || j == 3) {
                printf("[%d] ", road_networks[i][j]);
            } else {
                printf("%3d ", road_networks[i][j]);
            }
        }
        printf("\n");
    }

    // prompts user for origin point
    int origin;
    printf("Which point are you located? 0- A, 1 -B, 2-C...8 -I?\n");
    scanf("%d", &origin);

    printf("At point: %c\n", 'A' + origin);

    // if origin point happens to be exactly at c and d, print it is a charging station
    for (int i = 0; i < ROW; i++) {
      if (origin == 2){
        printf("C is a charging station.");
        break;
      }
      else if (origin ==3){
        printf("D is a charging station");
        break;
      }
      // if it is at other points, check if there is a direct path to the charging station
      else if (i == origin){
        for (int j = i; j <= COL; j++){
          if (road_networks[j][2]==1){
            printf("point: C arrived to charging station");
            break;
          }
          else if (road_networks[j][3]==1){
            printf("point: D arrived to charging station");
            break;
          }
          else {
            continue;
          }

        }
      }
```

OR



```c
#define COL 9

// initialize matrix
int main() {
    int road_networks[ROW][COL] = {
        {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}
    };

    // print matrix and add btackets for c and d
    printf("Adjacency matrix:\n");
    for (int i = 0; i < ROW; i++) {
        for (int j = 0; j < COL; j++) {
            if (i == 2 || j == 2 || i == 3 || j == 3) {
                printf("[%d] ", road_networks[i][j]);
            } else {
                printf("%3d ", road_networks[i][j]);
            }
        }
        printf("\n");
    }
  int origin;
  printf("Which point are you located? 0- A, 1 -B, 2-C...7 -H?\n");
  scanf("%d", &origin);
  switch(origin){
    case 0: case 1: case 5:
    printf("\nAt point: %c\npoint: C arrived to charging station", 'A' + origin);
    break;
    case 2:
    printf("\nAt point: %c\npoint: C is a charging station", 'A' + origin);
    break;
    case 3:
    printf("\n At point: D\npoint: D is a charging station", 'A' + origin);
    break;
    case 4: case 6:
    printf("\nAt point: E\npoint: D arrived to charging station", 'A' + origin);
    break;
    case 7:
    printf("\nAt point: H\npoint: No nearest charging station", 'A' + origin);
    break;
    default:
    printf("Choose only from Stations 1-7");
    break;
  }
  return 0;
}
```