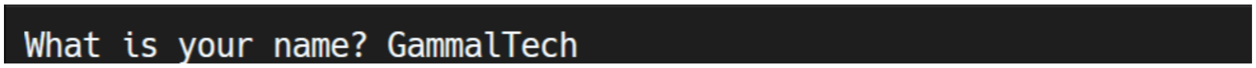
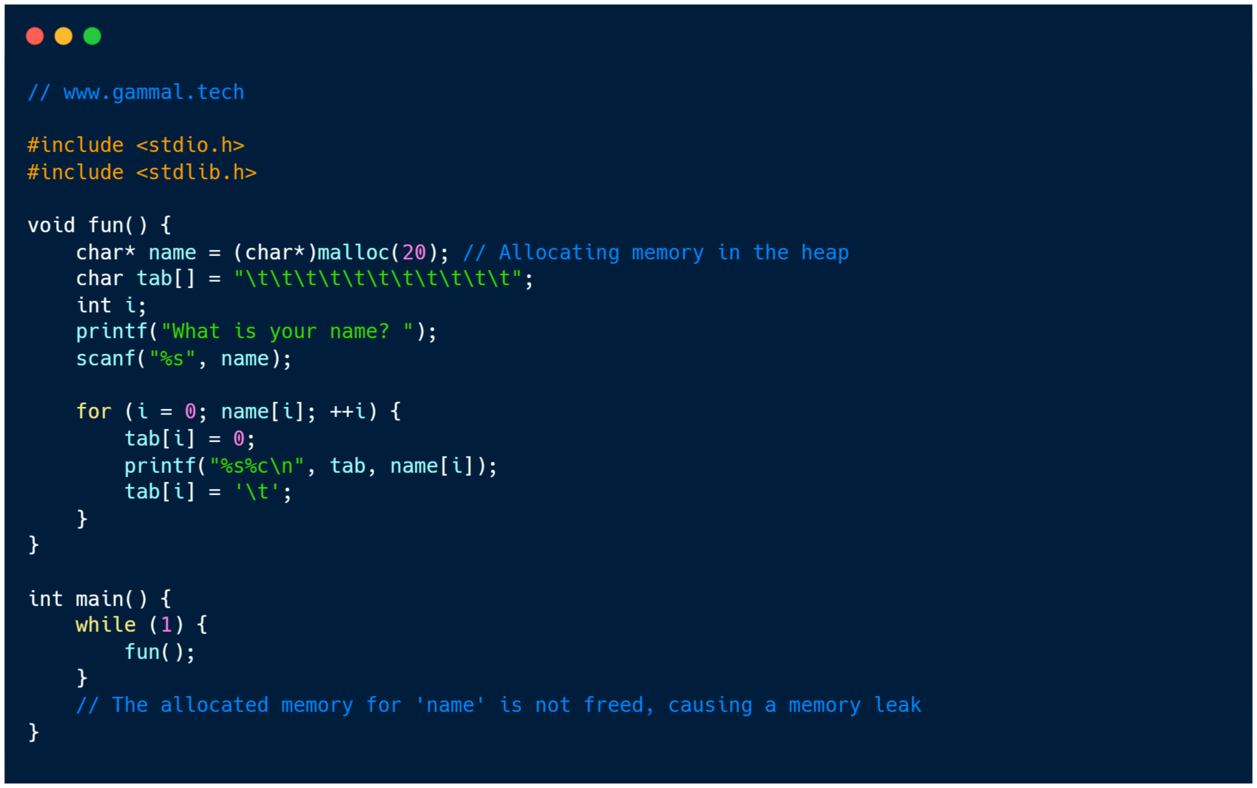
1- Trace the following program and predict the output.

Input

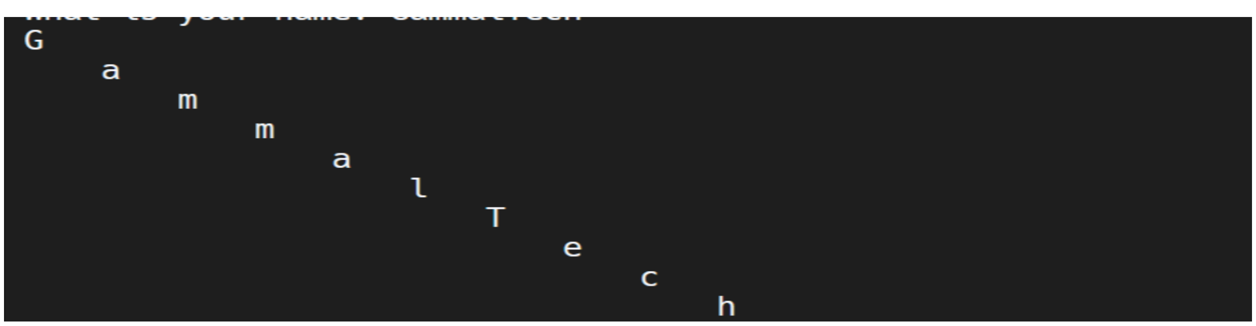


```
What is your name? GammalTech
```



```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

void fun() {
    char* name = (char*)malloc(20); // Allocating memory in the heap
    char tab[] = "\t\t\t\t\t\t\t\t\t\t\t\t";
    int i;
    printf("What is your name? ");
    scanf("%s", name);

    for (i = 0; name[i]; ++i) {
        tab[i] = 0;
        printf("%s%c\n", tab, name[i]);
        tab[i] = '\t';
    }
}

int main() {
    while (1) {
        fun();
    }
    // The allocated memory for 'name' is not freed, causing a memory leak
}
```

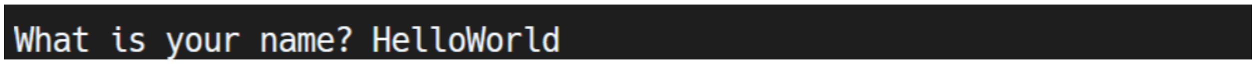
Solution



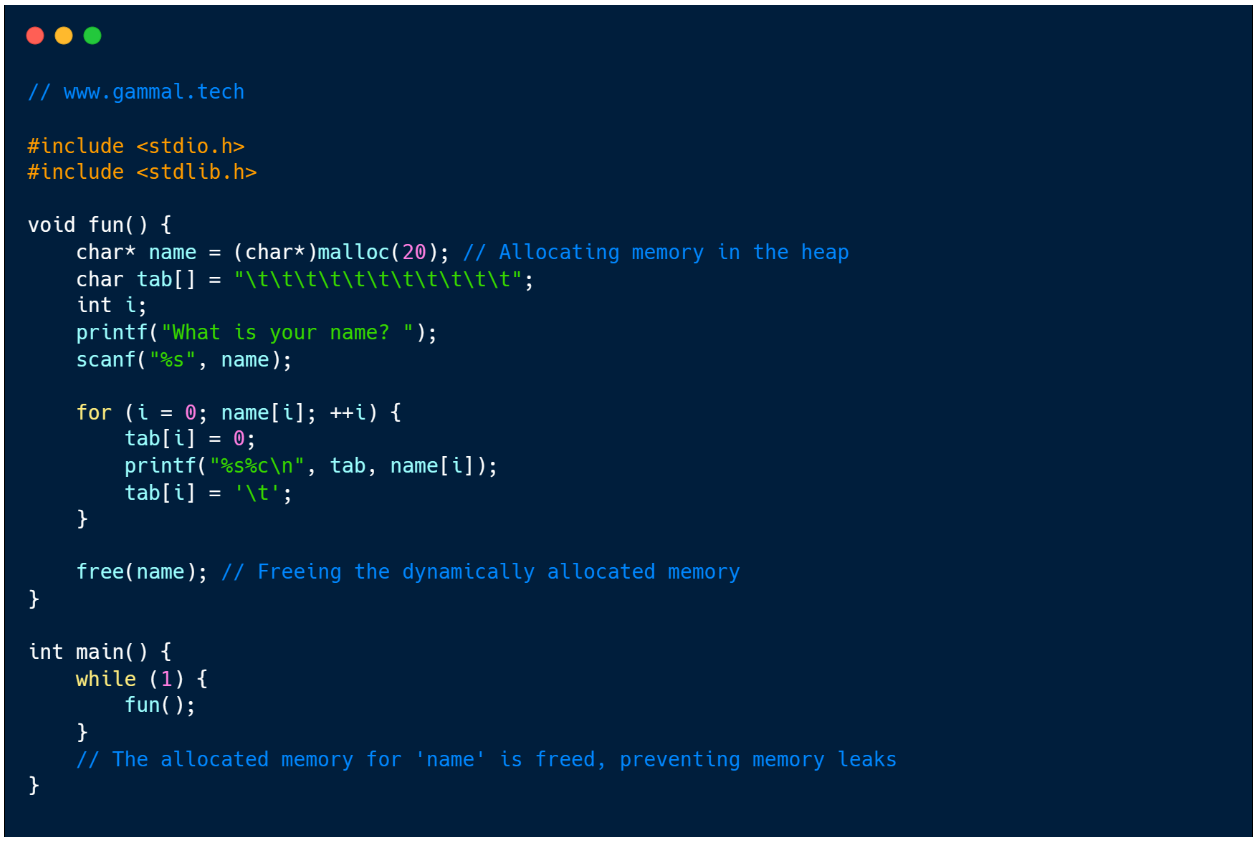
```
G
    a
        m
            m
                a
                    l
                        T
                            e
                                c
                                    h
```

2- Trace the following program and predict the output.

Input

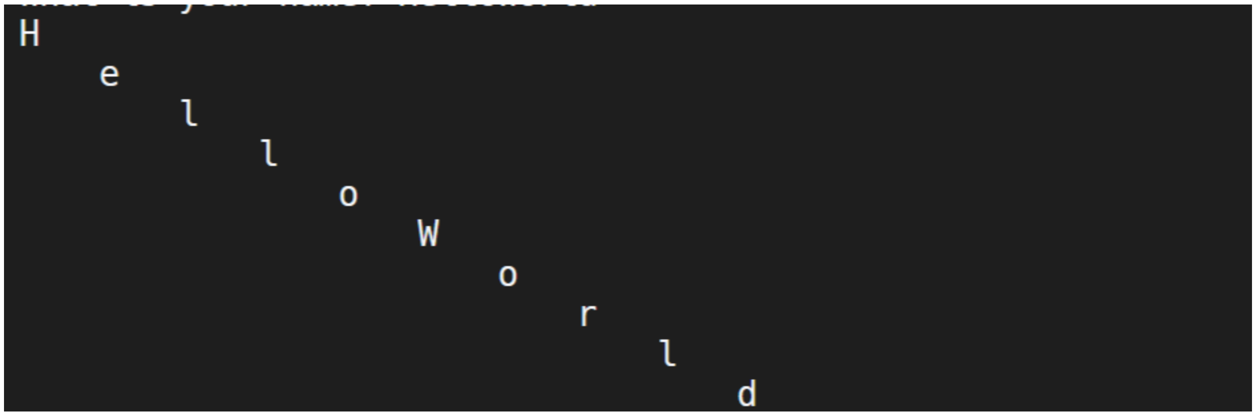


```
What is your name? HelloWorld
```



```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

void fun() {
    char* name = (char*)malloc(20); // Allocating memory in the heap
    char tab[] = "\t\t\t\t\t\t\t\t\t\t\t\t";
    int i;
    printf("What is your name? ");
    scanf("%s", name);

    for (i = 0; name[i]; ++i) {
        tab[i] = 0;
        printf("%s%c\n", tab, name[i]);
        tab[i] = '\t';
    }

    free(name); // Freeing the dynamically allocated memory
}

int main() {
    while (1) {
        fun();
    }
    // The allocated memory for 'name' is freed, preventing memory leaks
}
```

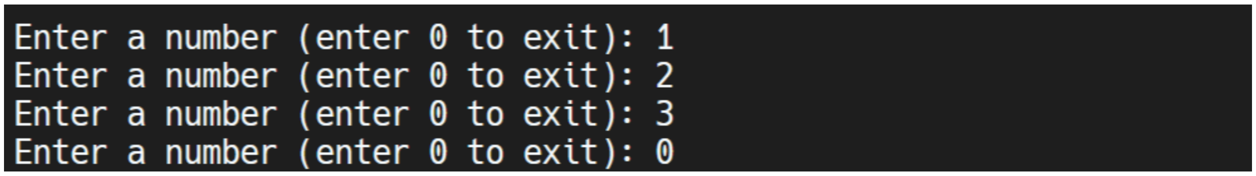
Solution



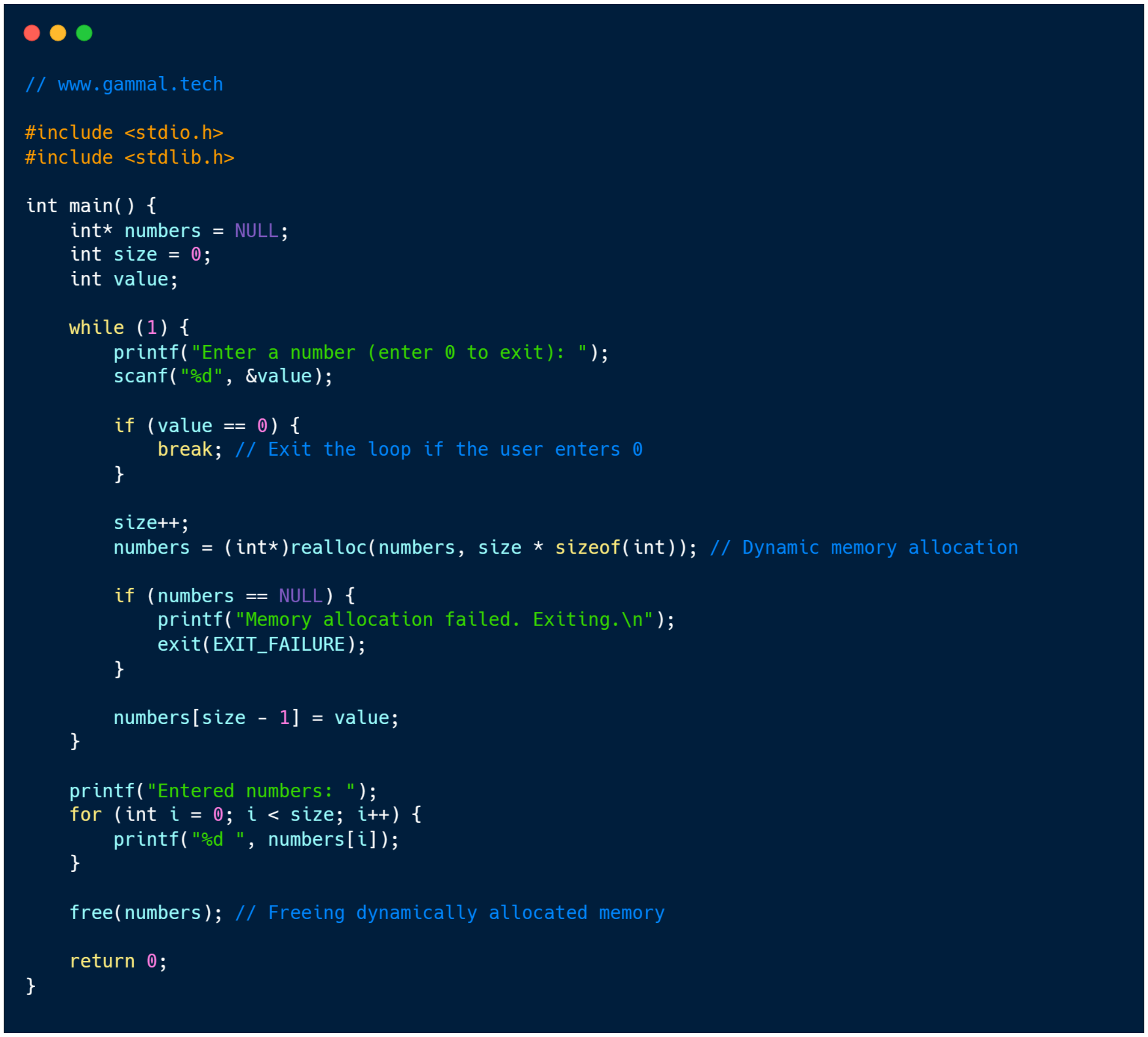
```
H
    e
        l
            l
                o
                    W
                        o
                            r
                                l
                                    d
```

3- Trace the following program and predict the output.

Input

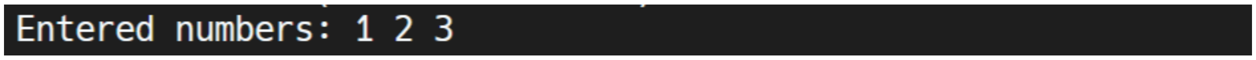


```
Enter a number (enter 0 to exit): 1
Enter a number (enter 0 to exit): 2
Enter a number (enter 0 to exit): 3
Enter a number (enter 0 to exit): 0
```



```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int main() {
    int* numbers = NULL;
    int size = 0;
    int value;

    while (1) {
        printf("Enter a number (enter 0 to exit): ");
        scanf("%d", &value);

        if (value == 0) {
            break; // Exit the loop if the user enters 0
        }

        size++;
        numbers = (int*)realloc(numbers, size * sizeof(int)); // Dynamic memory allocation

        if (numbers == NULL) {
            printf("Memory allocation failed. Exiting.\n");
            exit(EXIT_FAILURE);
        }

        numbers[size - 1] = value;
    }

    printf("Entered numbers: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", numbers[i]);
    }

    free(numbers); // Freeing dynamically allocated memory

    return 0;
}
```

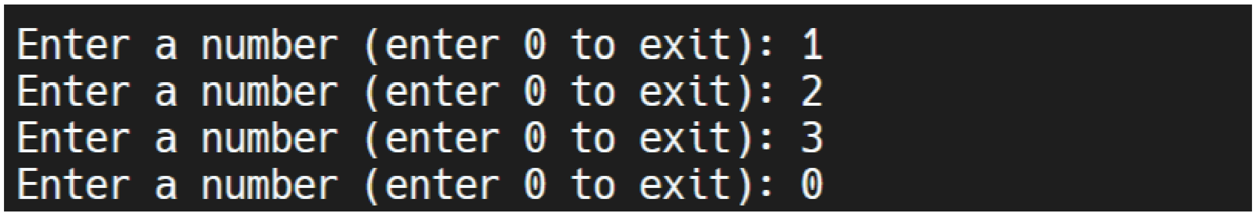
Solution



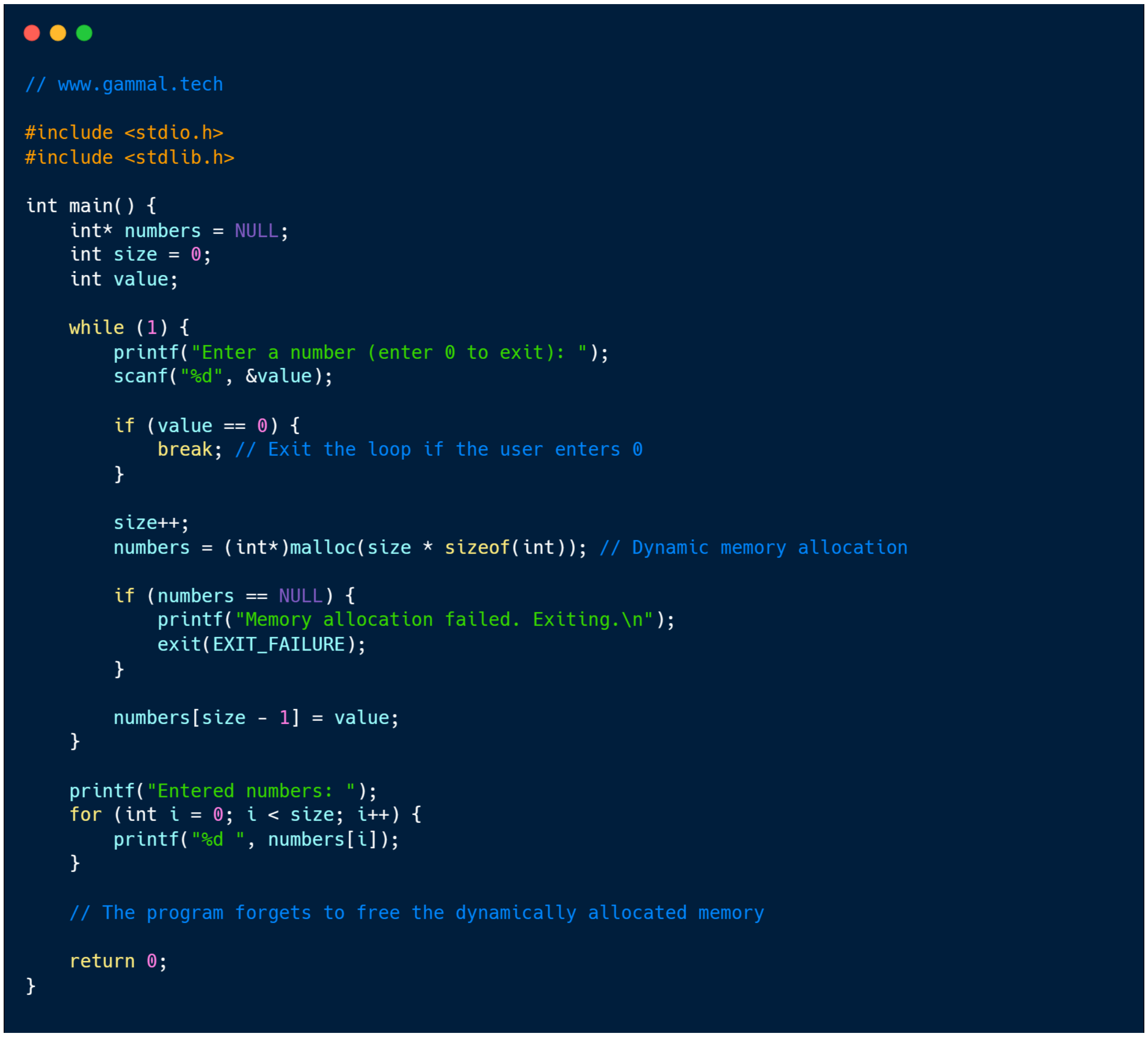
```
Entered numbers: 1 2 3
```

4- Trace the following program and predict the output.

Input

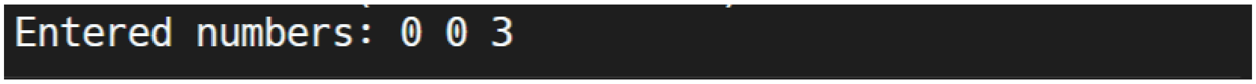


```
Enter a number (enter 0 to exit): 1
Enter a number (enter 0 to exit): 2
Enter a number (enter 0 to exit): 3
Enter a number (enter 0 to exit): 0
```



```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int main() {
    int* numbers = NULL;
    int size = 0;
    int value;

    while (1) {
        printf("Enter a number (enter 0 to exit): ");
        scanf("%d", &value);

        if (value == 0) {
            break; // Exit the loop if the user enters 0
        }

        size++;
        numbers = (int*)malloc(size * sizeof(int)); // Dynamic memory allocation

        if (numbers == NULL) {
            printf("Memory allocation failed. Exiting.\n");
            exit(EXIT_FAILURE);
        }

        numbers[size - 1] = value;
    }

    printf("Entered numbers: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", numbers[i]);
    }

    // The program forgets to free the dynamically allocated memory

    return 0;
}
```

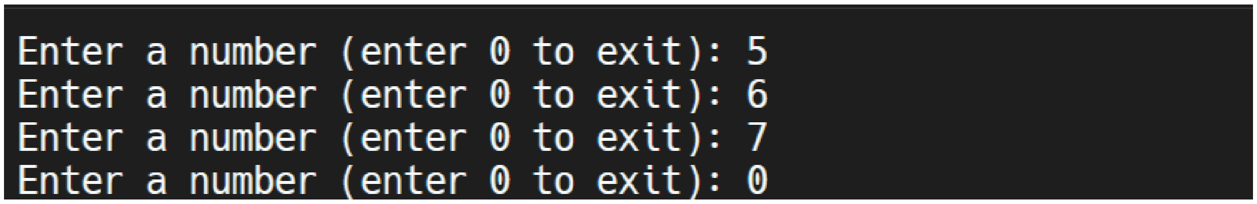
Solution



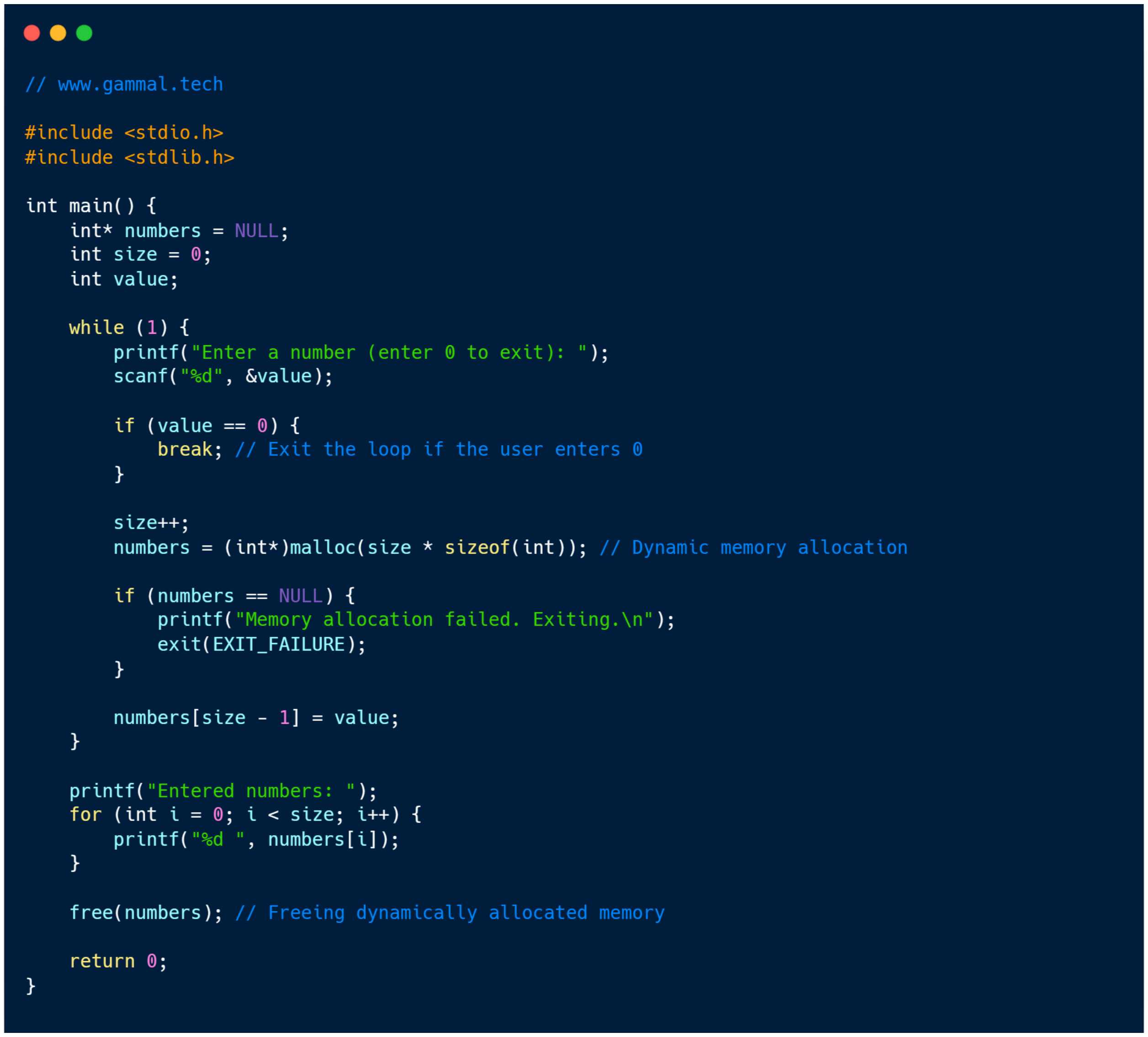
```
Entered numbers: 0 0 3
```

5- Trace the following program and predict the output.

Input

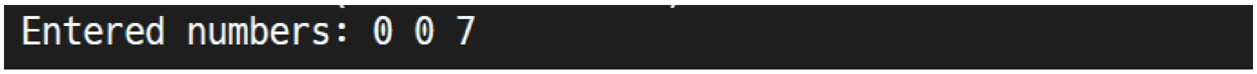


```
Enter a number (enter 0 to exit): 5
Enter a number (enter 0 to exit): 6
Enter a number (enter 0 to exit): 7
Enter a number (enter 0 to exit): 0
```



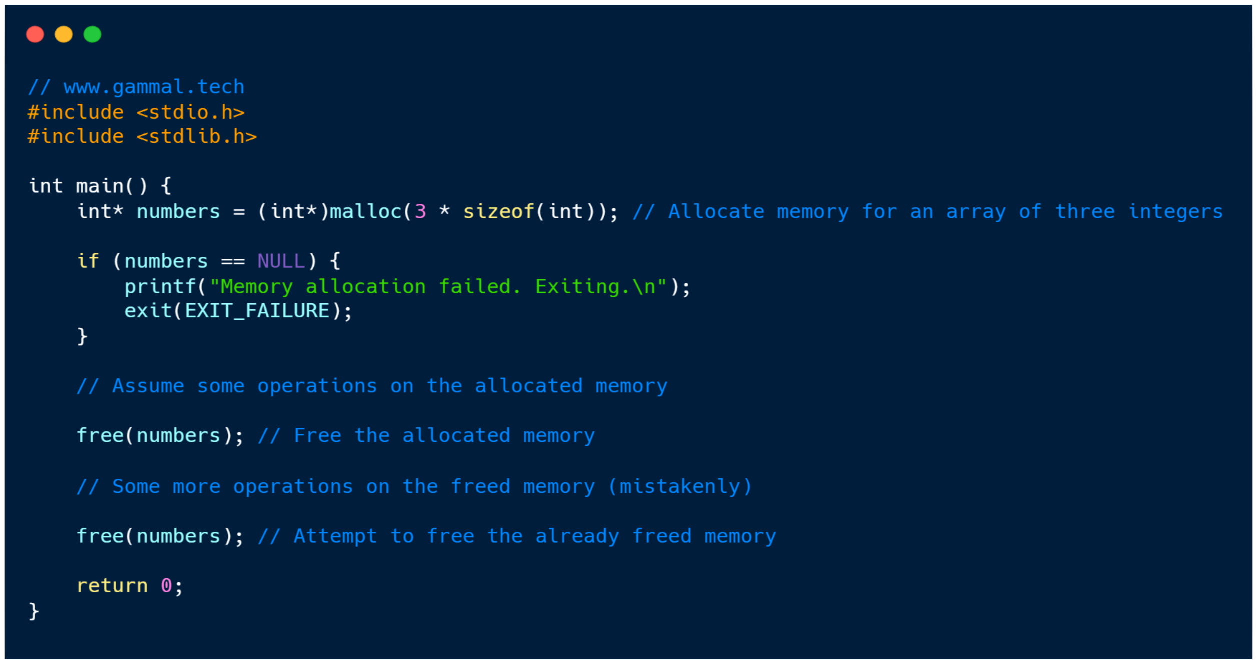
```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int main() {
    int* numbers = NULL;
    int size = 0;
    int value;

    while (1) {
        printf("Enter a number (enter 0 to exit): ");
        scanf("%d", &value);

        if (value == 0) {
            break; // Exit the loop if the user enters 0
        }

        size++;
        numbers = (int*)malloc(size * sizeof(int)); // Dynamic memory allocation

        if (numbers == NULL) {
            printf("Memory allocation failed. Exiting.\n");
            exit(EXIT_FAILURE);
        }

        numbers[size - 1] = value;
    }

    printf("Entered numbers: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", numbers[i]);
    }

    free(numbers); // Freeing dynamically allocated memory

    return 0;
}
```

Solution



```
Entered numbers: 0 0 7
```

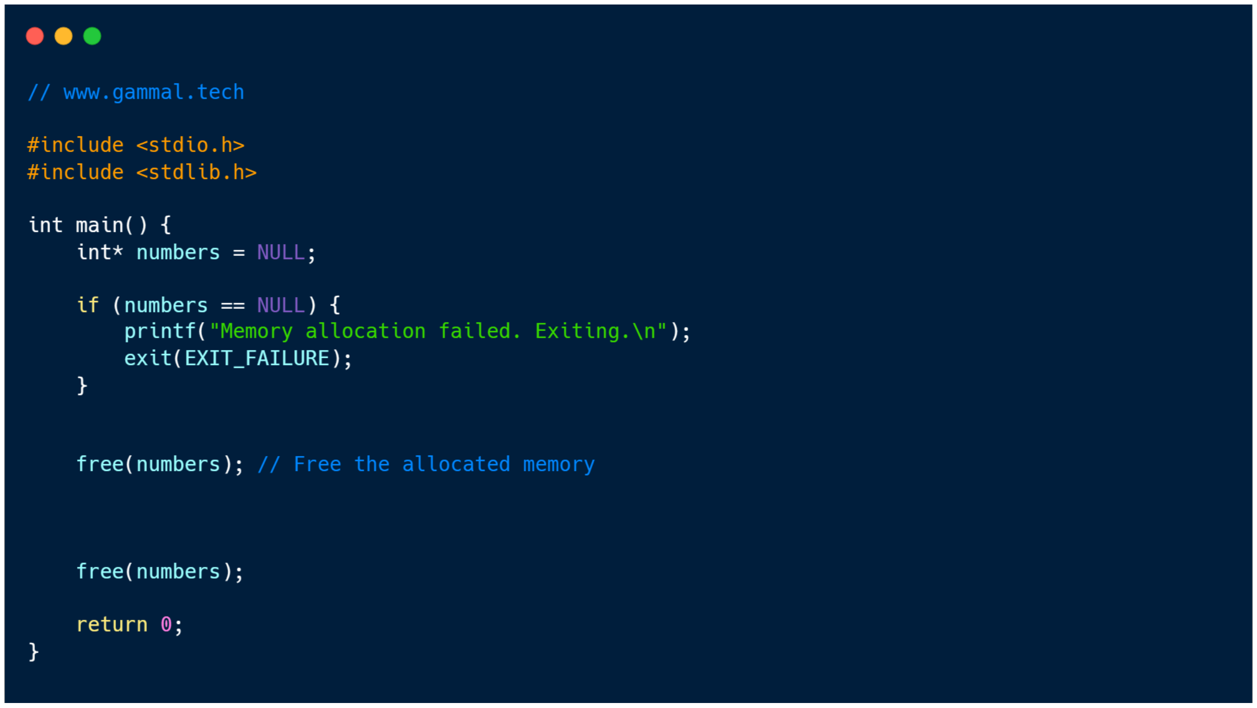
6- Trace the following program and predict the output.



```c
// www.gammal.tech
#include <stdio.h>
#include <stdlib.h>

int main() {
    int* numbers = (int*)malloc(3 * sizeof(int)); // Allocate memory for an array of three integers

    if (numbers == NULL) {
        printf("Memory allocation failed. Exiting.\n");
        exit(EXIT_FAILURE);
    }

    // Assume some operations on the allocated memory

    free(numbers); // Free the allocated memory

    // Some more operations on the freed memory (mistakenly)

    free(numbers); // Attempt to free the already freed memory

    return 0;
}
```

Solution

7- Trace the following program and predict the output.

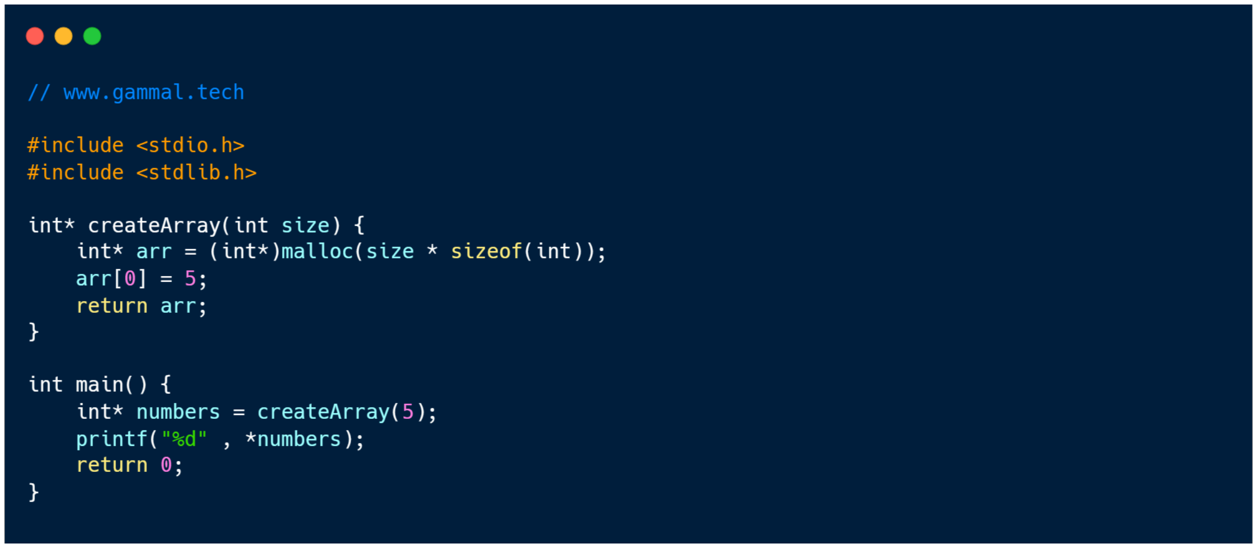


```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int main() {
    int* numbers = NULL;

    if (numbers == NULL) {
        printf("Memory allocation failed. Exiting.\n");
        exit(EXIT_FAILURE);
    }


    free(numbers); // Free the allocated memory



    free(numbers);

    return 0;
}
```

Solution

```
Memory allocation failed. Exiting.
```

8- Trace the following program and predict the output.

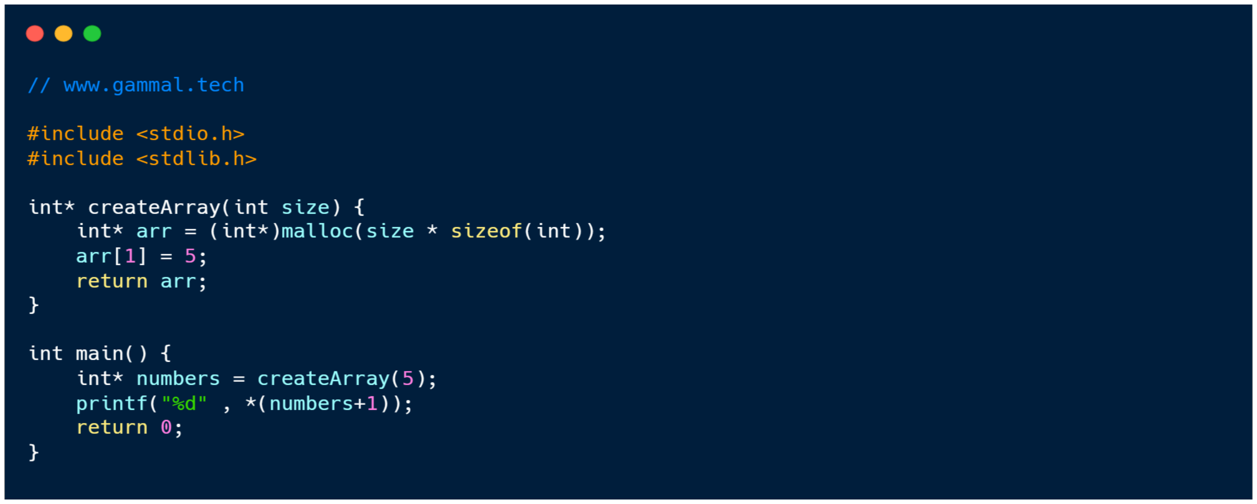


```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[0] = 5;
    return arr;
}

int main() {
    int* numbers = createArray(5);
    printf("%d" , *numbers);
    return 0;
}
```

Solution

```
5
```

9- Trace the following program and predict the output.

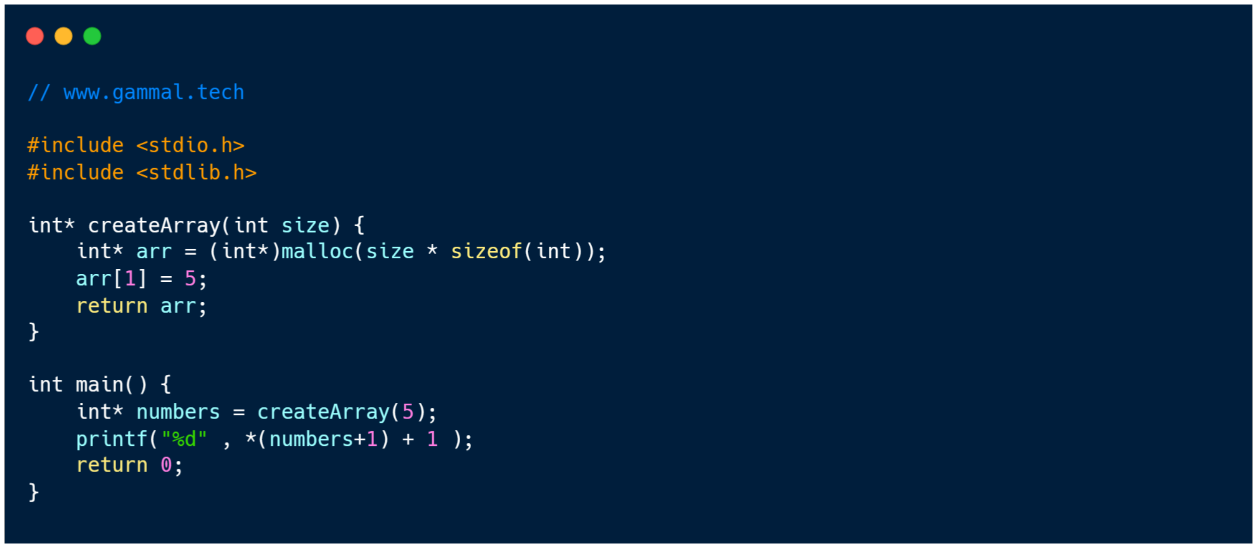


```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[1] = 5;
    return arr;
}

int main() {
    int* numbers = createArray(5);
    printf("%d" , *(numbers+1));
    return 0;
}
```

Solution

```
5
```

10- Trace the following program and predict the output.



```c
// www.gammal.tech

#include <stdio.h>
#include <stdlib.h>

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[1] = 5;
    return arr;
}

int main() {
    int* numbers = createArray(5);
    printf("%d" , *(numbers+1) + 1 );
    return 0;
}
```

Solution

```
6
```
