1- 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;
    arr[1] = 6;
    return arr;
}

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

Solution

```
5 6
```

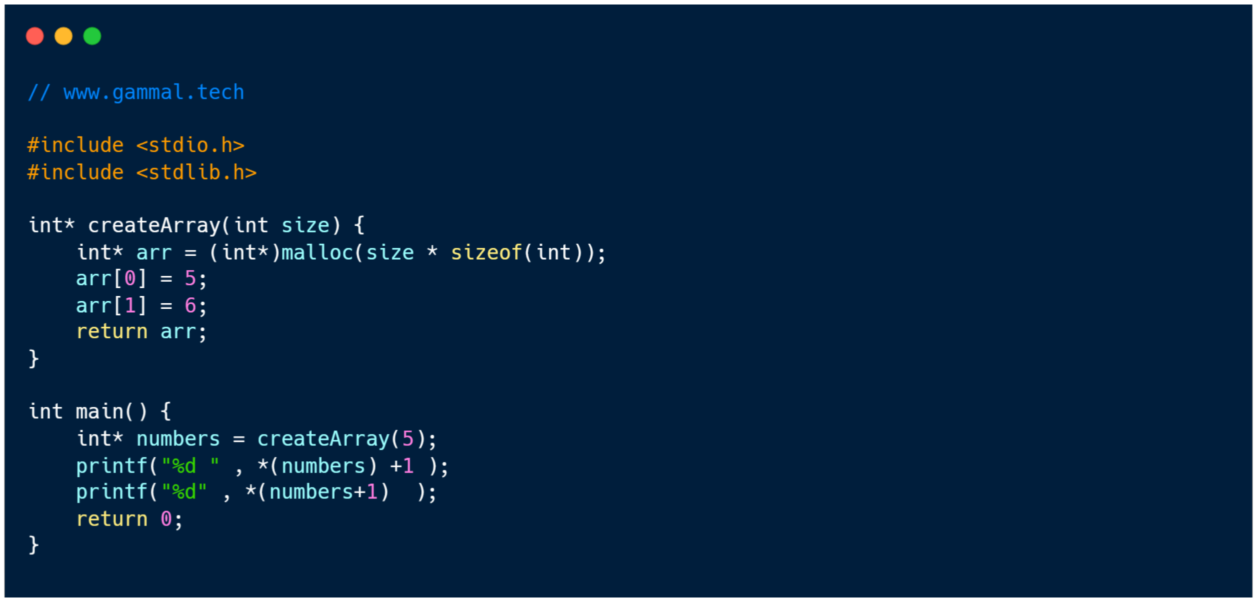
2- Trace the following program and predict the output.

```c
// www.gammal.tech

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

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[0] = 5;
    arr[1] = 6;
    return arr;
}

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

Solution

```
6 7
```

3- Trace the following program and predict the output.

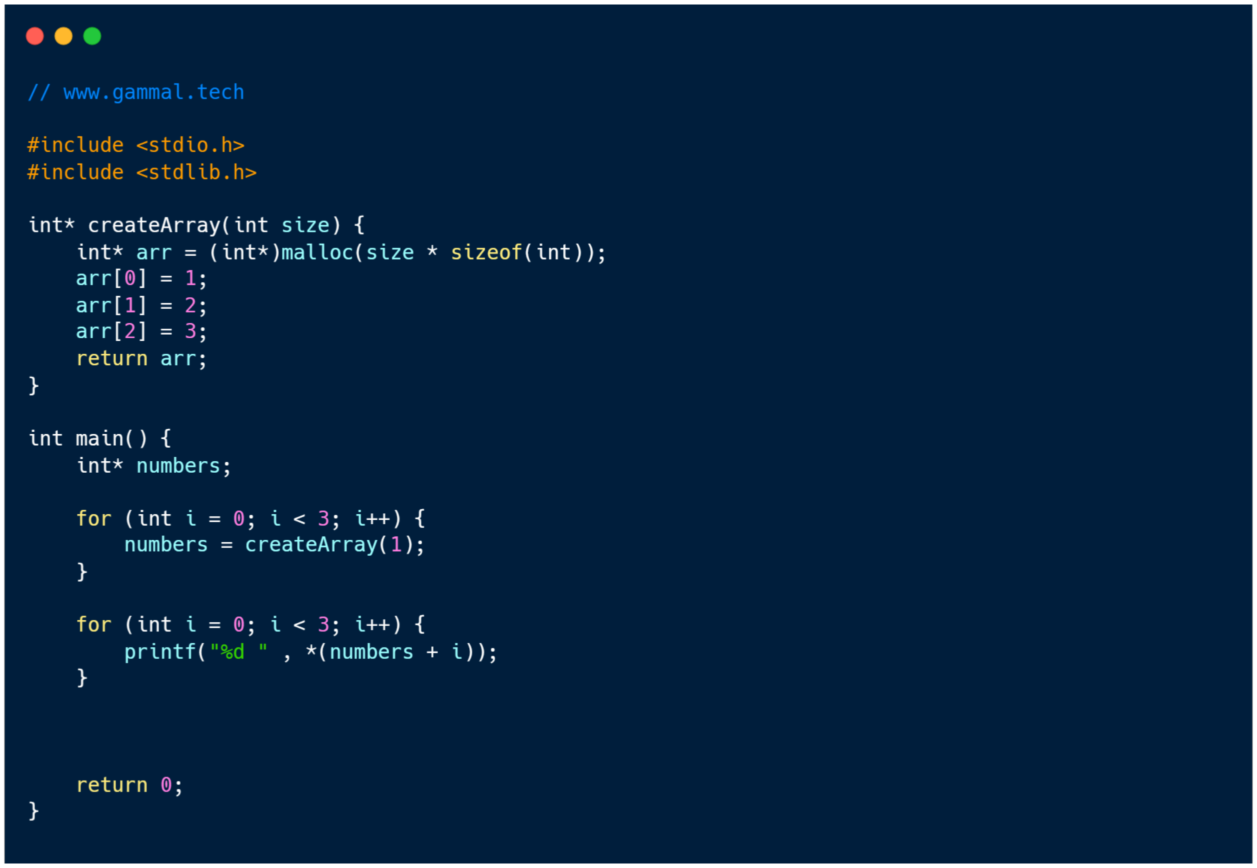


```c
// www.gammal.tech

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

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[0] = 5;
    arr[1] = 6;
    return arr;
}

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

Solution

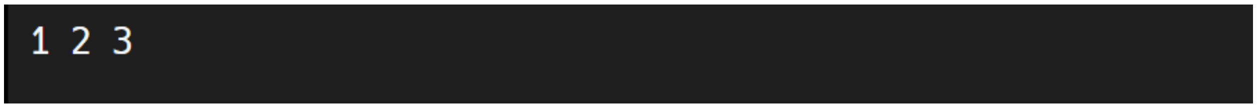
```
6 6
```

4- Trace the following program and predict the output.



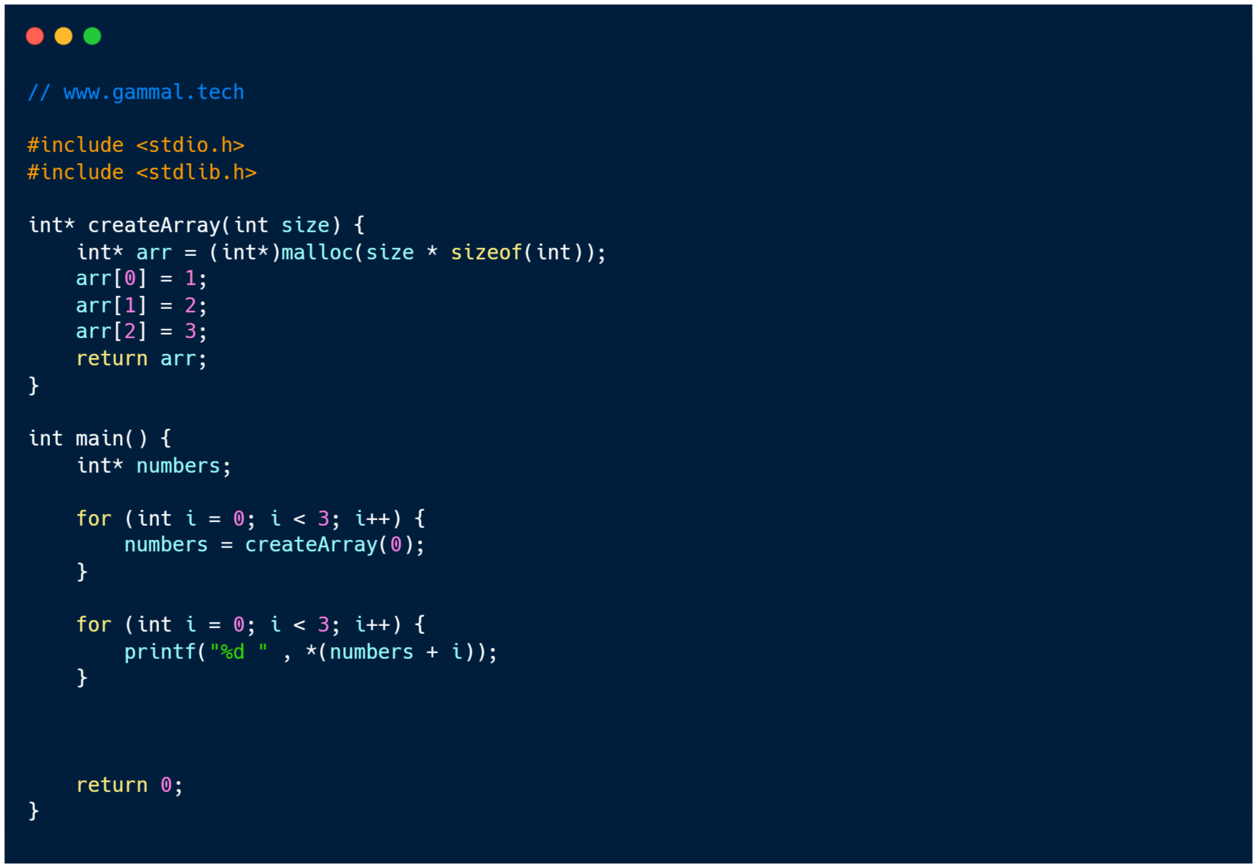
```c
// www.gammal.tech

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

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[0] = 1;
    arr[1] = 2;
    arr[2] = 3;
    return arr;
}

int main() {
    int* numbers;

    for (int i = 0; i < 3; i++) {
        numbers = createArray(1);
    }

    for (int i = 0; i < 3; i++) {
        printf("%d " , *(numbers + i));
    }



    return 0;
}
```

Solution



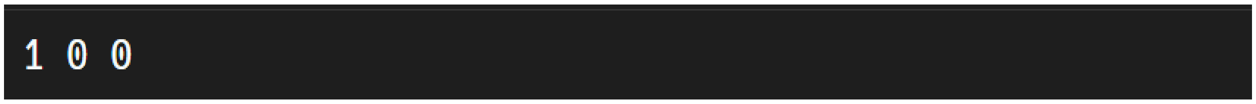
```
1 2 3
```

5- Trace the following program and predict the output.



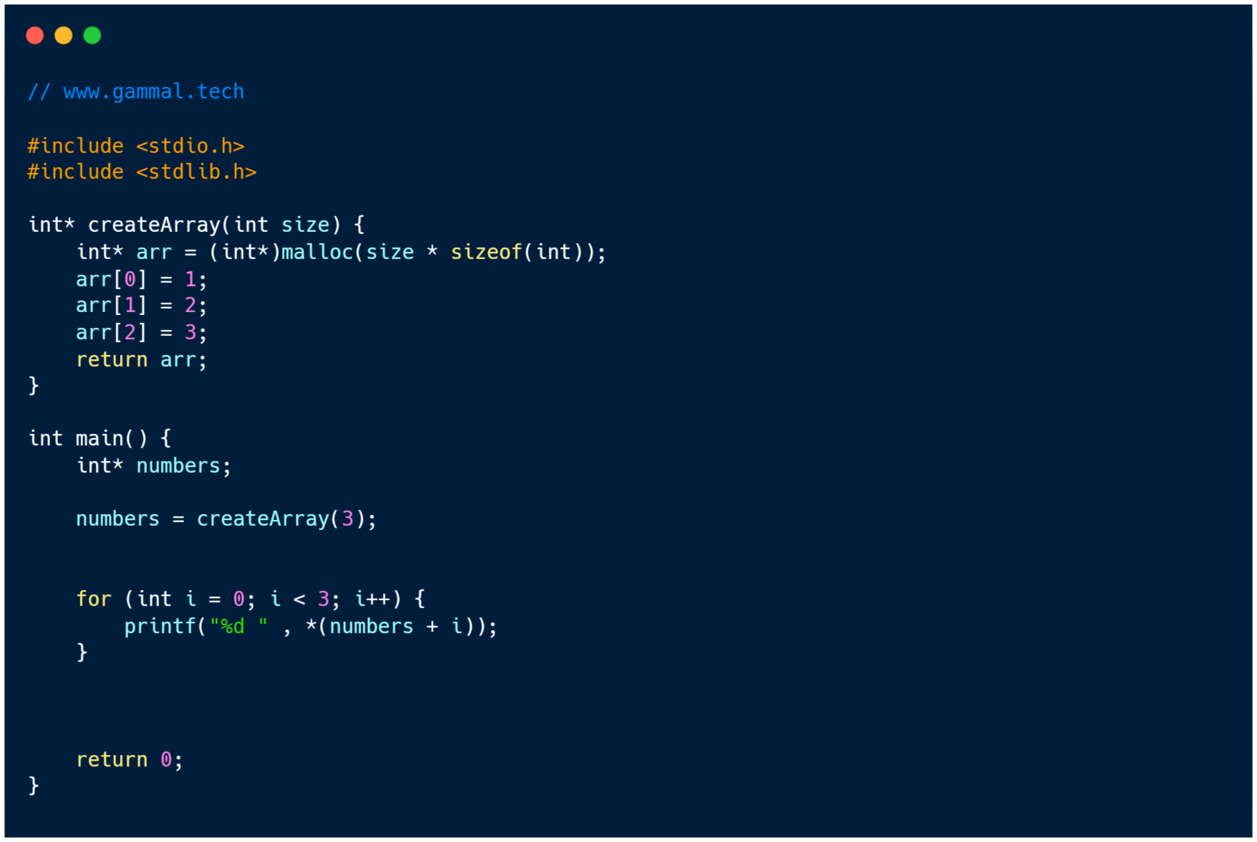
```c
// www.gammal.tech

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

int* createArray(int size) {
    int* arr = (int*)malloc(size * sizeof(int));
    arr[0] = 1;
    arr[1] = 2;
    arr[2] = 3;
    return arr;
}

int main() {
    int* numbers;

    for (int i = 0; i < 3; i++) {
        numbers = createArray(0);
    }

    for (int i = 0; i < 3; i++) {
        printf("%d " , *(numbers + i));
    }



    return 0;
}
```

Solution



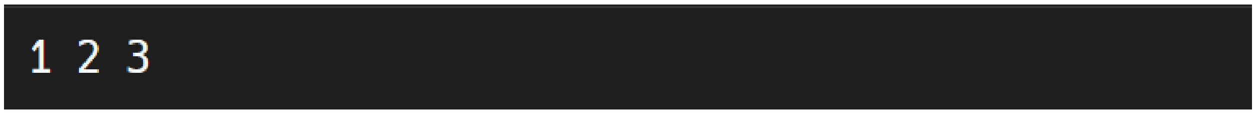
```
1 0 0
```

6- 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] = 1;
    arr[1] = 2;
    arr[2] = 3;
    return arr;
}

int main() {
    int* numbers;

    numbers = createArray(3);


    for (int i = 0; i < 3; i++) {
        printf("%d " , *(numbers + i));
    }



    return 0;
}
```

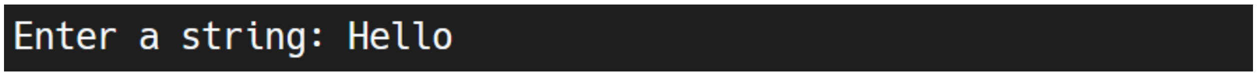
Solution



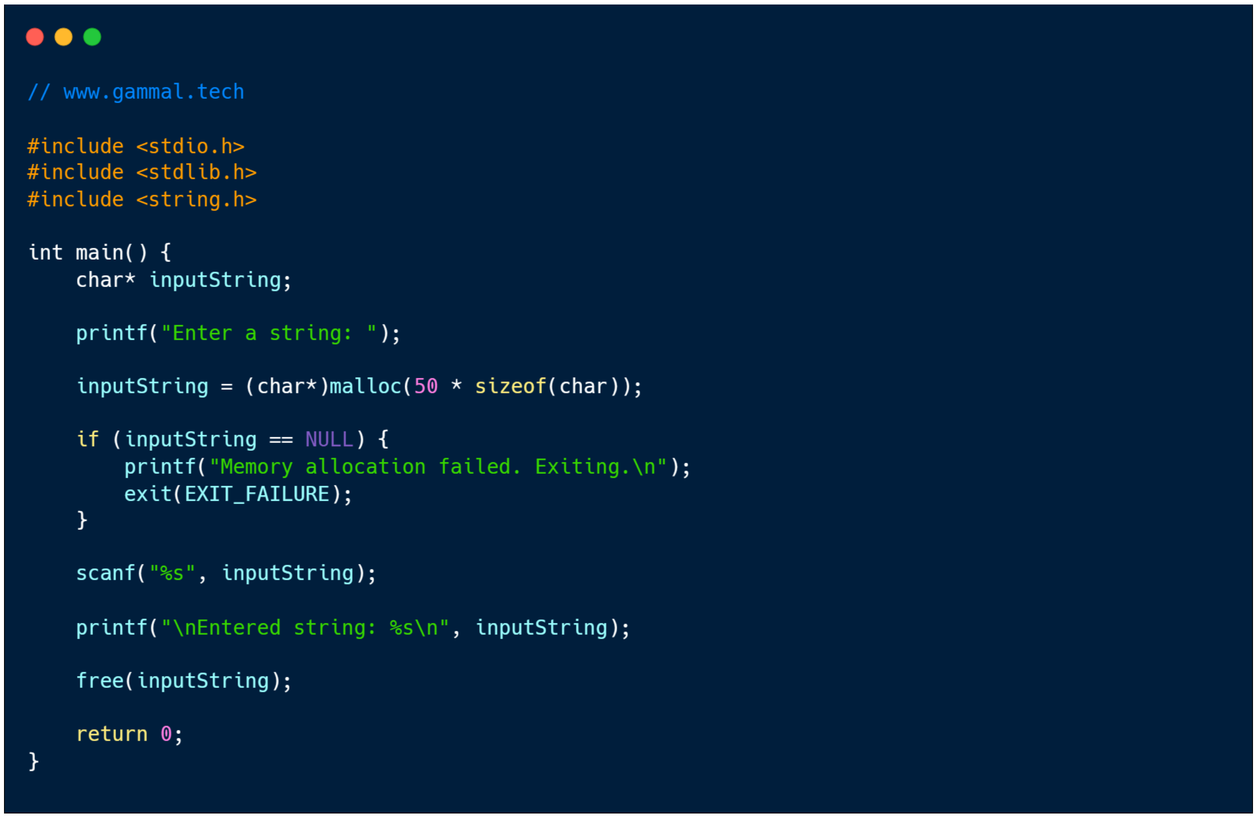
```
1 2 3
```

7- Trace the following program and predict the output.

Input

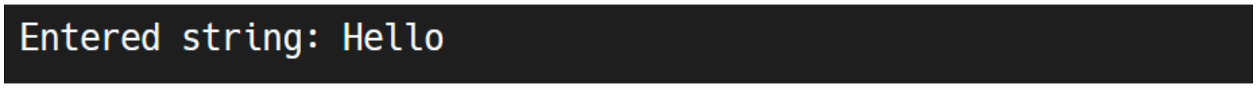


```
Enter a string: Hello
```



```c
// www.gammal.tech

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

int main() {
    char* inputString;

    printf("Enter a string: ");

    inputString = (char*)malloc(50 * sizeof(char));

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

    scanf("%s", inputString);

    printf("\nEntered string: %s\n", inputString);

    free(inputString);

    return 0;
}
```

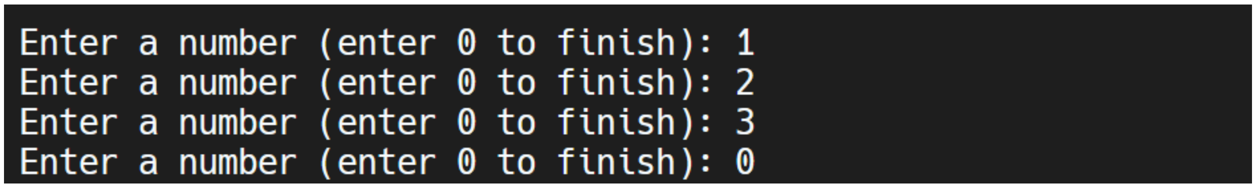
Solution



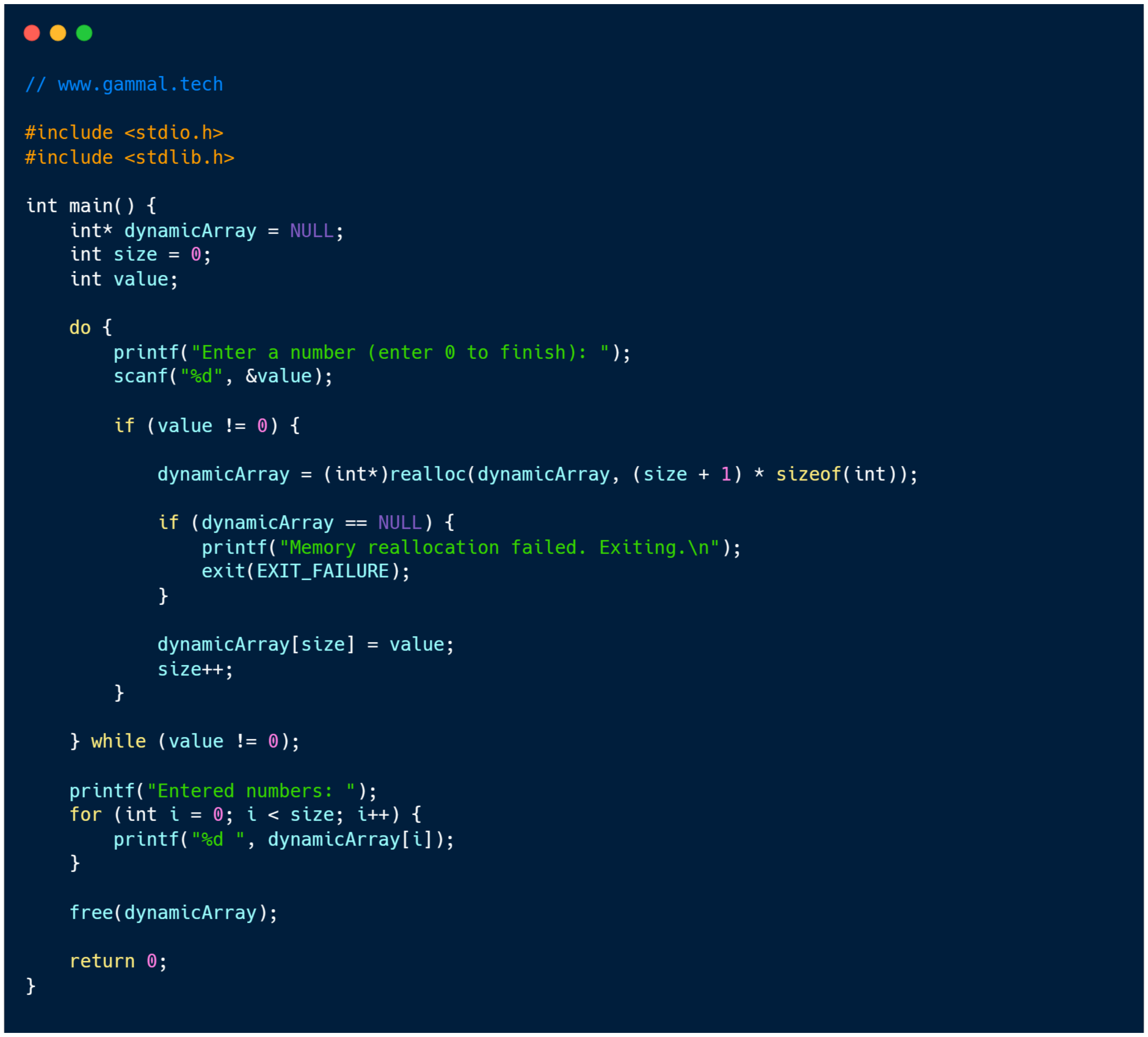
```
Entered string: Hello
```

8- Trace the following program and predict the output.

Input

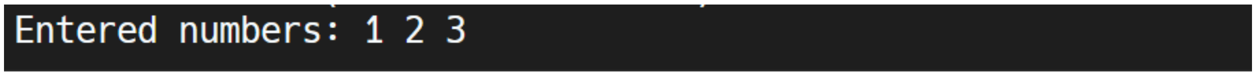


```
Enter a number (enter 0 to finish): 1
Enter a number (enter 0 to finish): 2
Enter a number (enter 0 to finish): 3
Enter a number (enter 0 to finish): 0
```



```c
// www.gammal.tech

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

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

    do {
        printf("Enter a number (enter 0 to finish): ");
        scanf("%d", &value);

        if (value != 0) {

            dynamicArray = (int*)realloc(dynamicArray, (size + 1) * sizeof(int));

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

            dynamicArray[size] = value;
            size++;
        }

    } while (value != 0);

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

    free(dynamicArray);

    return 0;
}
```

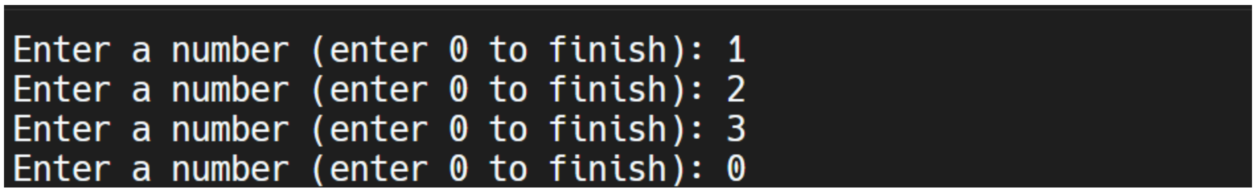
Output



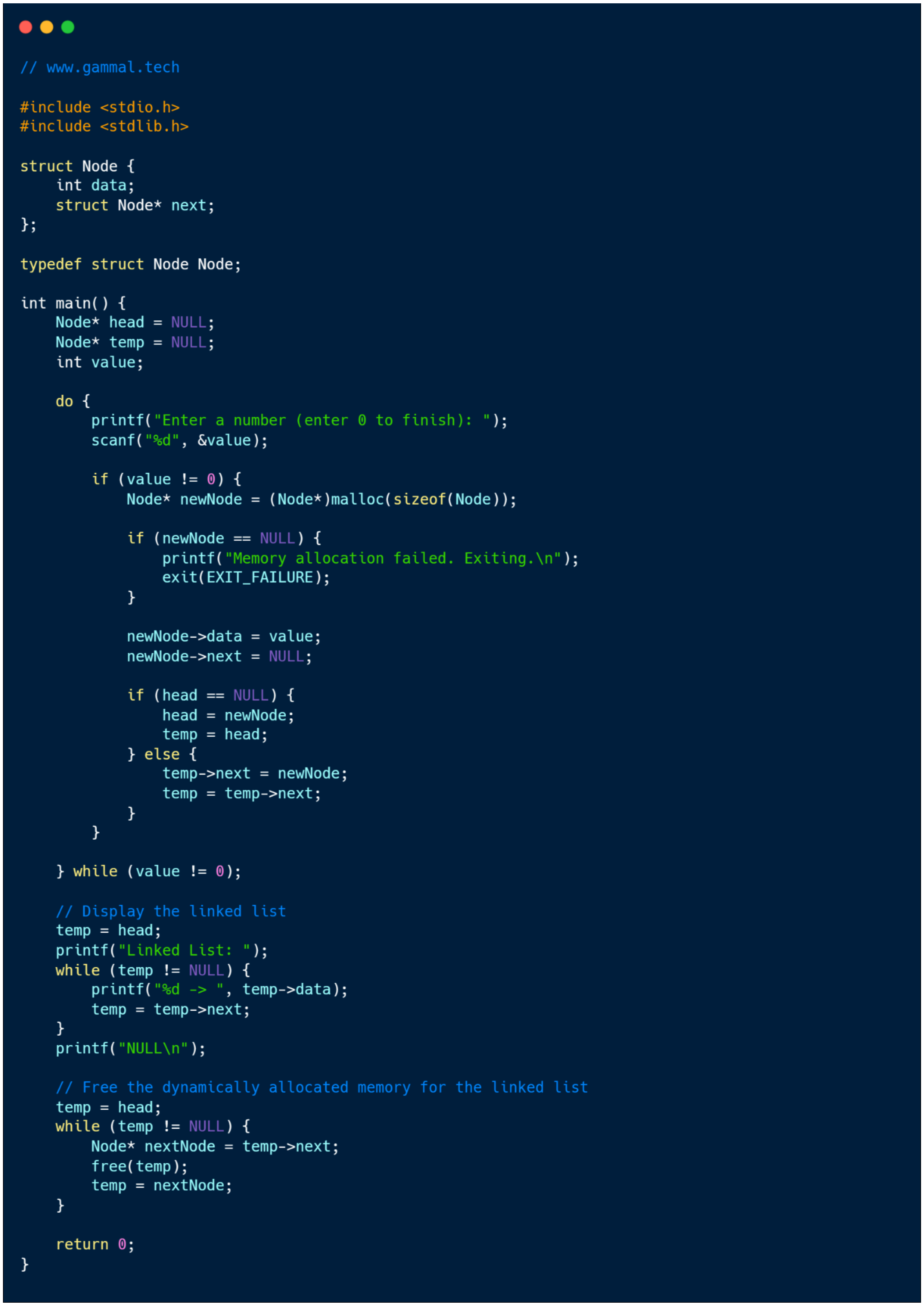
```
Entered numbers: 1 2 3
```

9- Trace the following program and predict the output.

Input

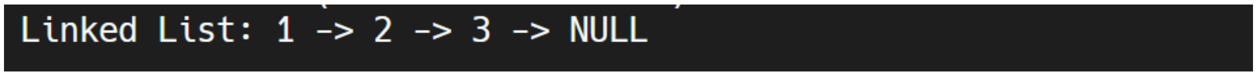


```
Enter a number (enter 0 to finish): 1
Enter a number (enter 0 to finish): 2
Enter a number (enter 0 to finish): 3
Enter a number (enter 0 to finish): 0
```



```c
// www.gammal.tech

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

struct Node {
    int data;
    struct Node* next;
};

typedef struct Node Node;

int main() {
    Node* head = NULL;
    Node* temp = NULL;
    int value;

    do {
        printf("Enter a number (enter 0 to finish): ");
        scanf("%d", &value);

        if (value != 0) {
            Node* newNode = (Node*)malloc(sizeof(Node));

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

            newNode->data = value;
            newNode->next = NULL;

            if (head == NULL) {
                head = newNode;
                temp = head;
            } else {
                temp->next = newNode;
                temp = temp->next;
            }
        }

    } while (value != 0);

    // Display the linked list
    temp = head;
    printf("Linked List: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");

    // Free the dynamically allocated memory for the linked list
    temp = head;
    while (temp != NULL) {
        Node* nextNode = temp->next;
        free(temp);
        temp = nextNode;
    }

    return 0;
}
```

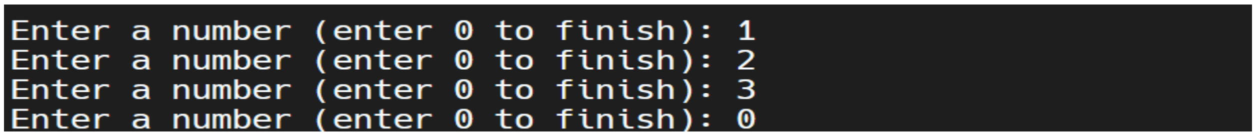
Solution



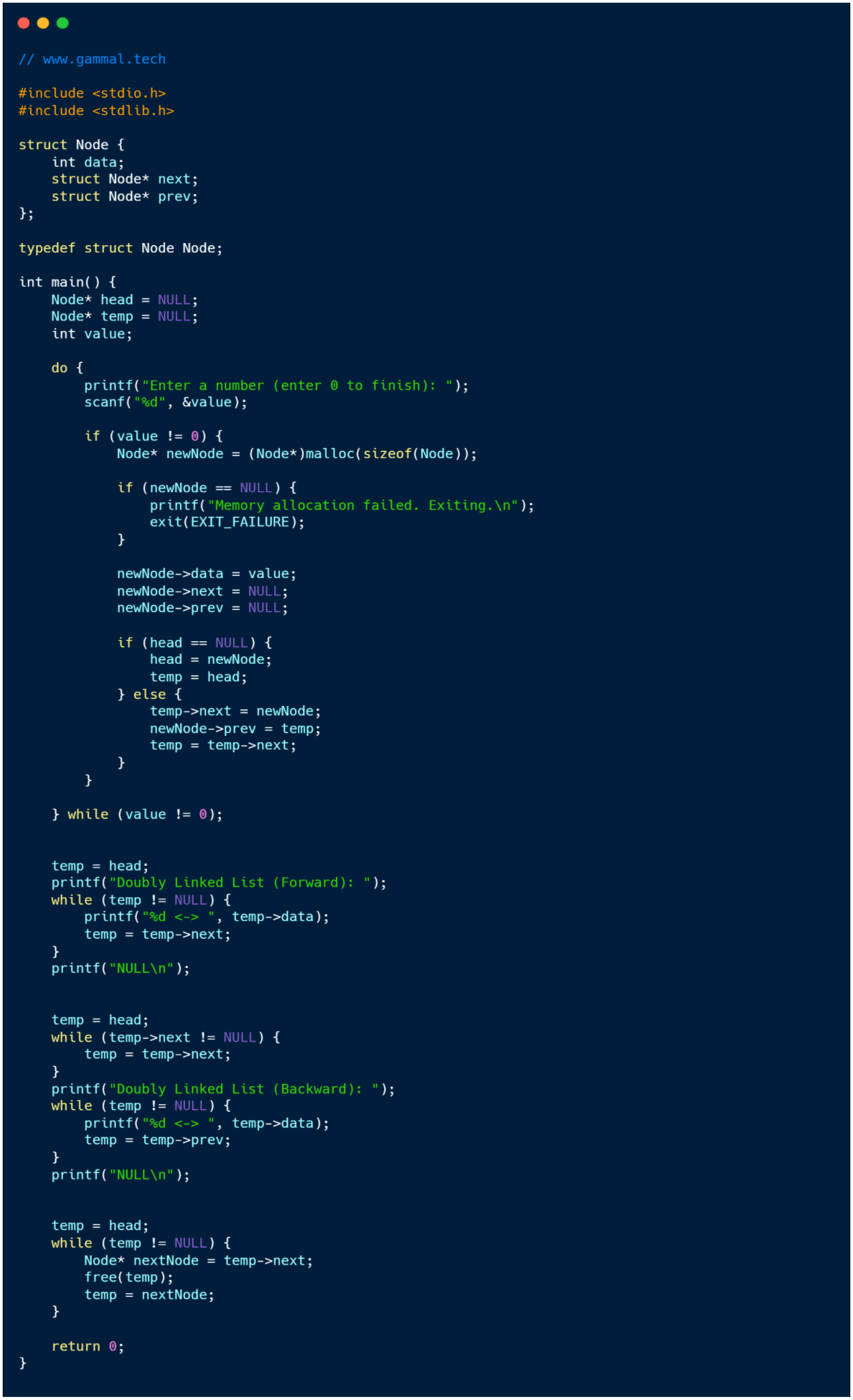
```
Linked List: 1 -> 2 -> 3 -> NULL
```

10- Trace the following program and predict the output.

Input

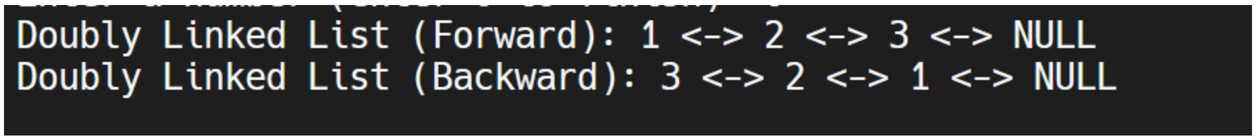


```
Enter a number (enter 0 to finish): 1
Enter a number (enter 0 to finish): 2
Enter a number (enter 0 to finish): 3
Enter a number (enter 0 to finish): 0
```



```c
// www.gammal.tech

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

struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

typedef struct Node Node;

int main() {
    Node* head = NULL;
    Node* temp = NULL;
    int value;

    do {
        printf("Enter a number (enter 0 to finish): ");
        scanf("%d", &value);

        if (value != 0) {
            Node* newNode = (Node*)malloc(sizeof(Node));

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

            newNode->data = value;
            newNode->next = NULL;
            newNode->prev = NULL;

            if (head == NULL) {
                head = newNode;
                temp = head;
            } else {
                temp->next = newNode;
                newNode->prev = temp;
                temp = temp->next;
            }
        }

    } while (value != 0);


    temp = head;
    printf("Doubly Linked List (Forward): ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");


    temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    printf("Doubly Linked List (Backward): ");
    while (temp != NULL) {
        printf("%d <-> ", temp->data);
        temp = temp->prev;
    }
    printf("NULL\n");


    temp = head;
    while (temp != NULL) {
        Node* nextNode = temp->next;
        free(temp);
        temp = nextNode;
    }

    return 0;
}
```

Solution



```
Doubly Linked List (Forward): 1 <-> 2 <-> 3 <-> NULL
Doubly Linked List (Backward): 3 <-> 2 <-> 1 <-> NULL
```