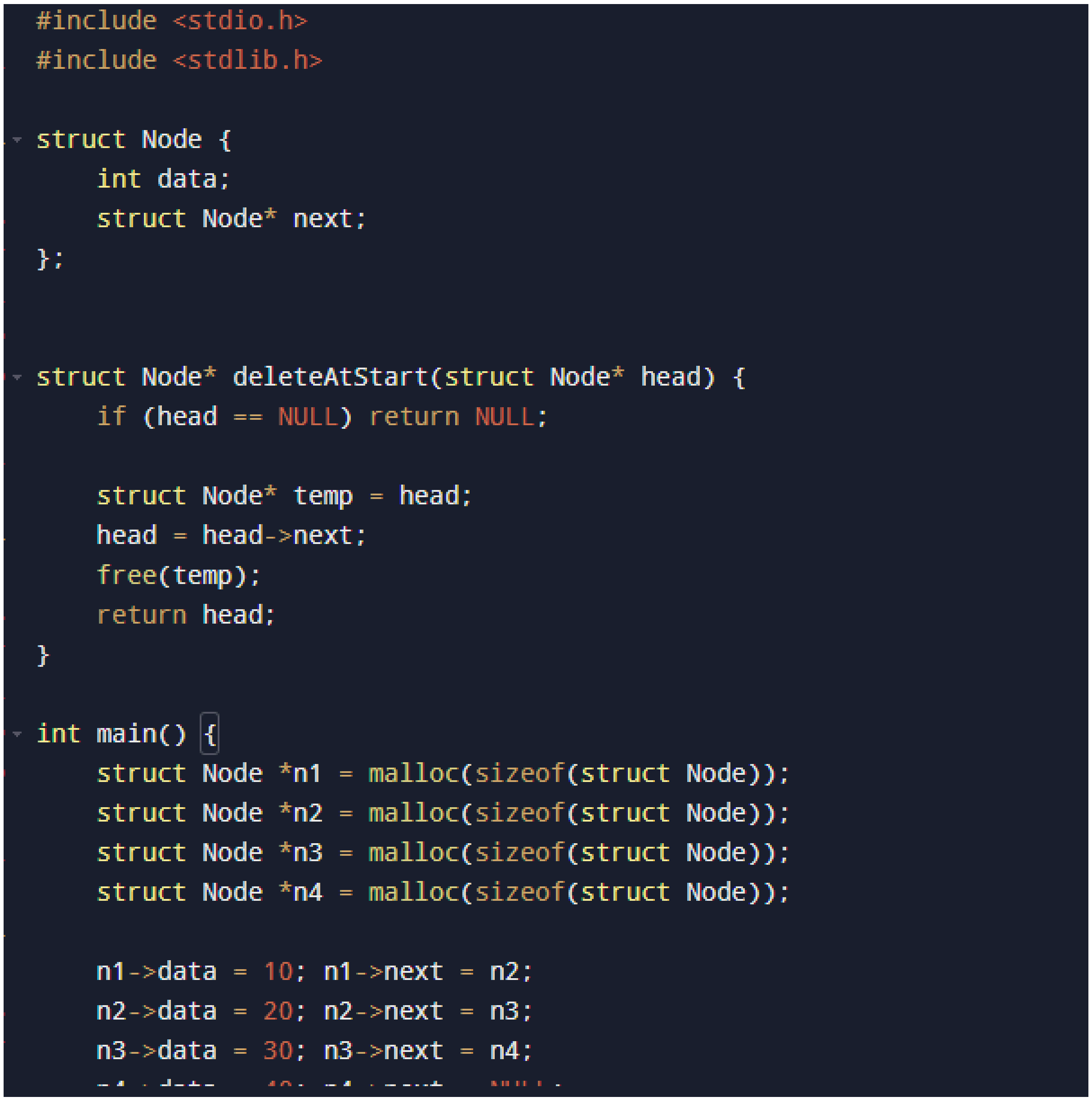
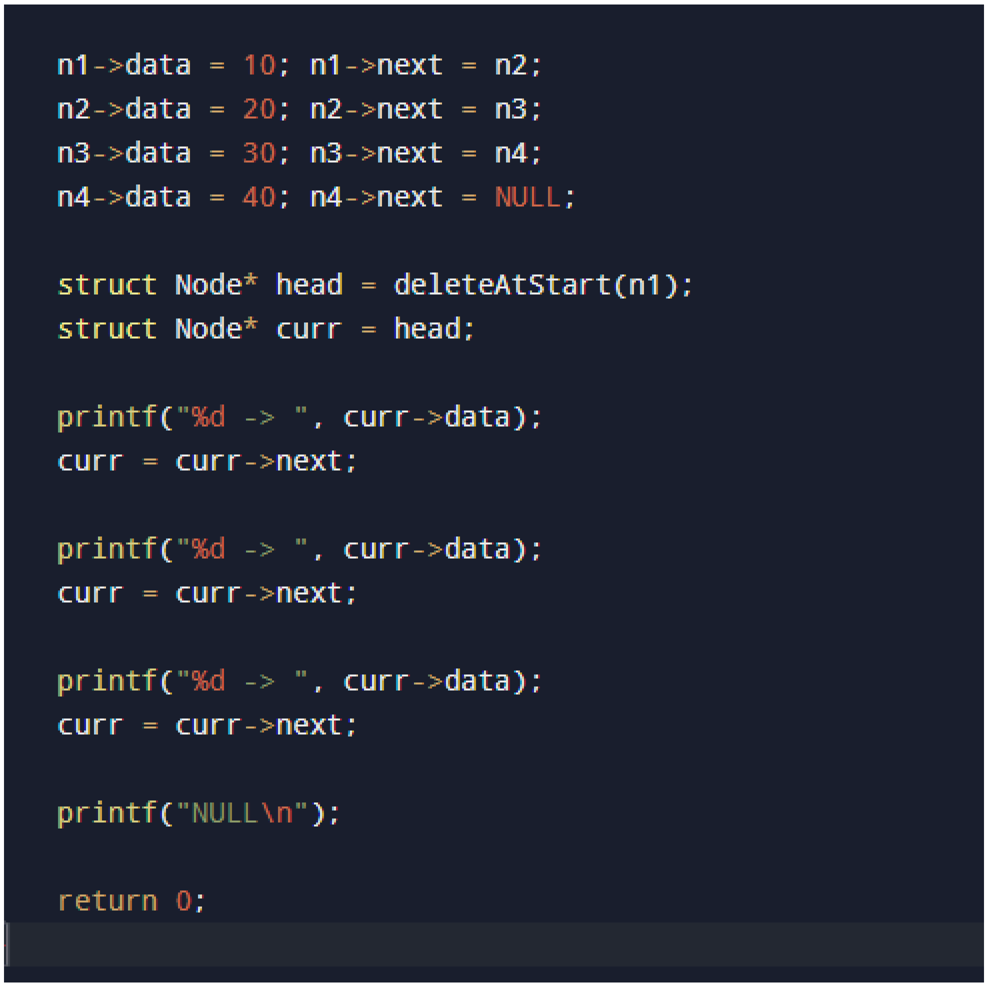
1RUA25BCA0096 Sanjana Pal

Lab 5
Deletion at Start
Input:



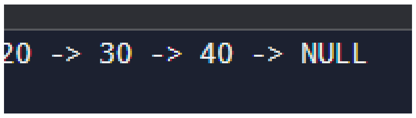
```c
#include <stdio.h>
#include <stdlib.h>

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


struct Node* deleteAtStart(struct Node* head) {
    if (head == NULL) return NULL;

    struct Node* temp = head;
    head = head->next;
    free(temp);
    return head;
}

int main() {
    struct Node *n1 = malloc(sizeof(struct Node));
    struct Node *n2 = malloc(sizeof(struct Node));
    struct Node *n3 = malloc(sizeof(struct Node));
    struct Node *n4 = malloc(sizeof(struct Node));

    n1->data = 10; n1->next = n2;
    n2->data = 20; n2->next = n3;
    n3->data = 30; n3->next = n4;
    n4->data = 40; n4->next = NULL;
```

1RUA25BCA0096 Sanjana Pal



```
n1->data = 10; n1->next = n2;
n2->data = 20; n2->next = n3;
n3->data = 30; n3->next = n4;
n4->data = 40; n4->next = NULL;

struct Node* head = deleteAtStart(n1);
struct Node* curr = head;

printf("%d -> ", curr->data);
curr = curr->next;

printf("%d -> ", curr->data);
curr = curr->next;

printf("%d -> ", curr->data);
curr = curr->next;

printf("NULL\n");

return 0;
```

Output:

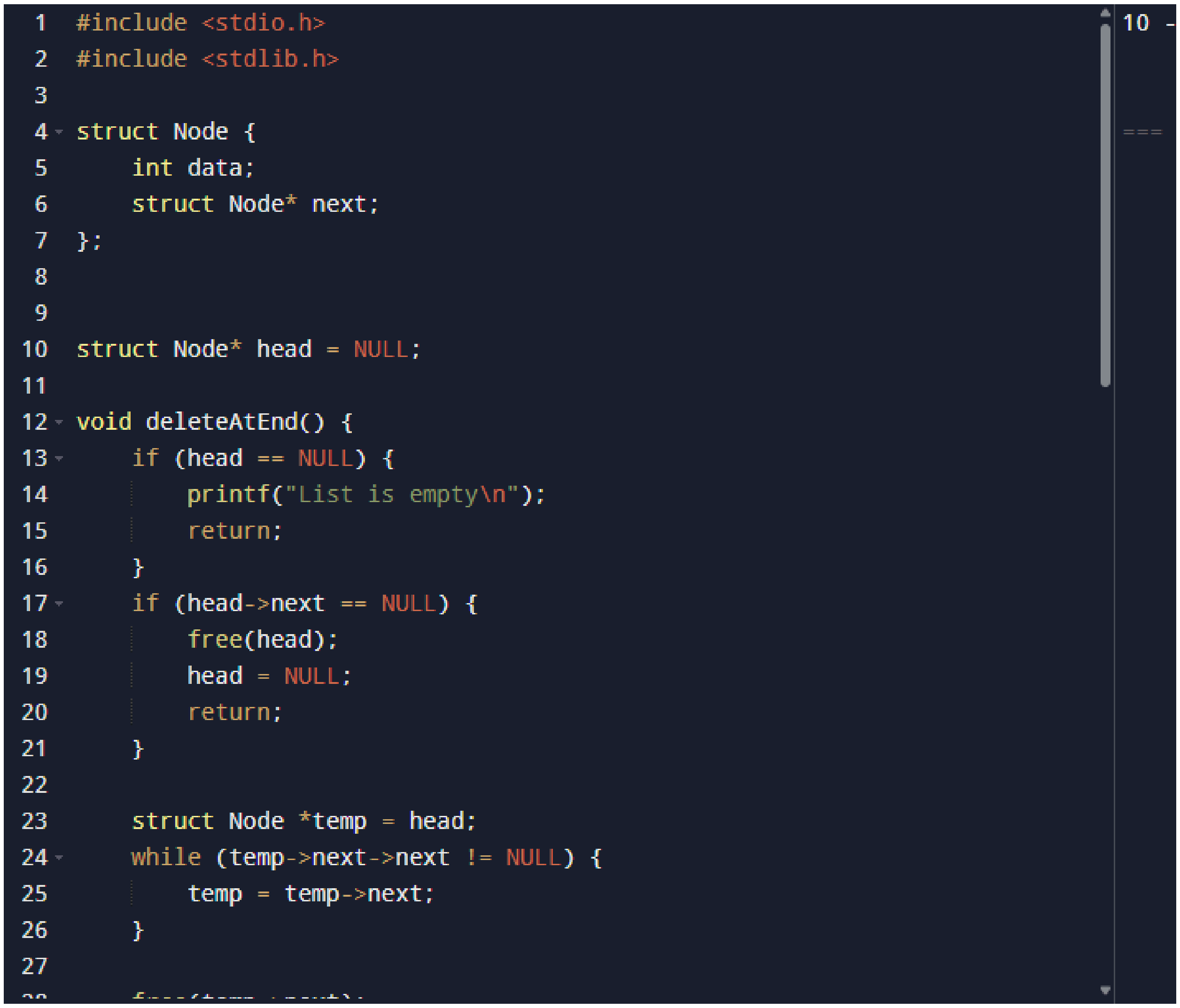


```
20 -> 30 -> 40 -> NULL
```

Deletion at End

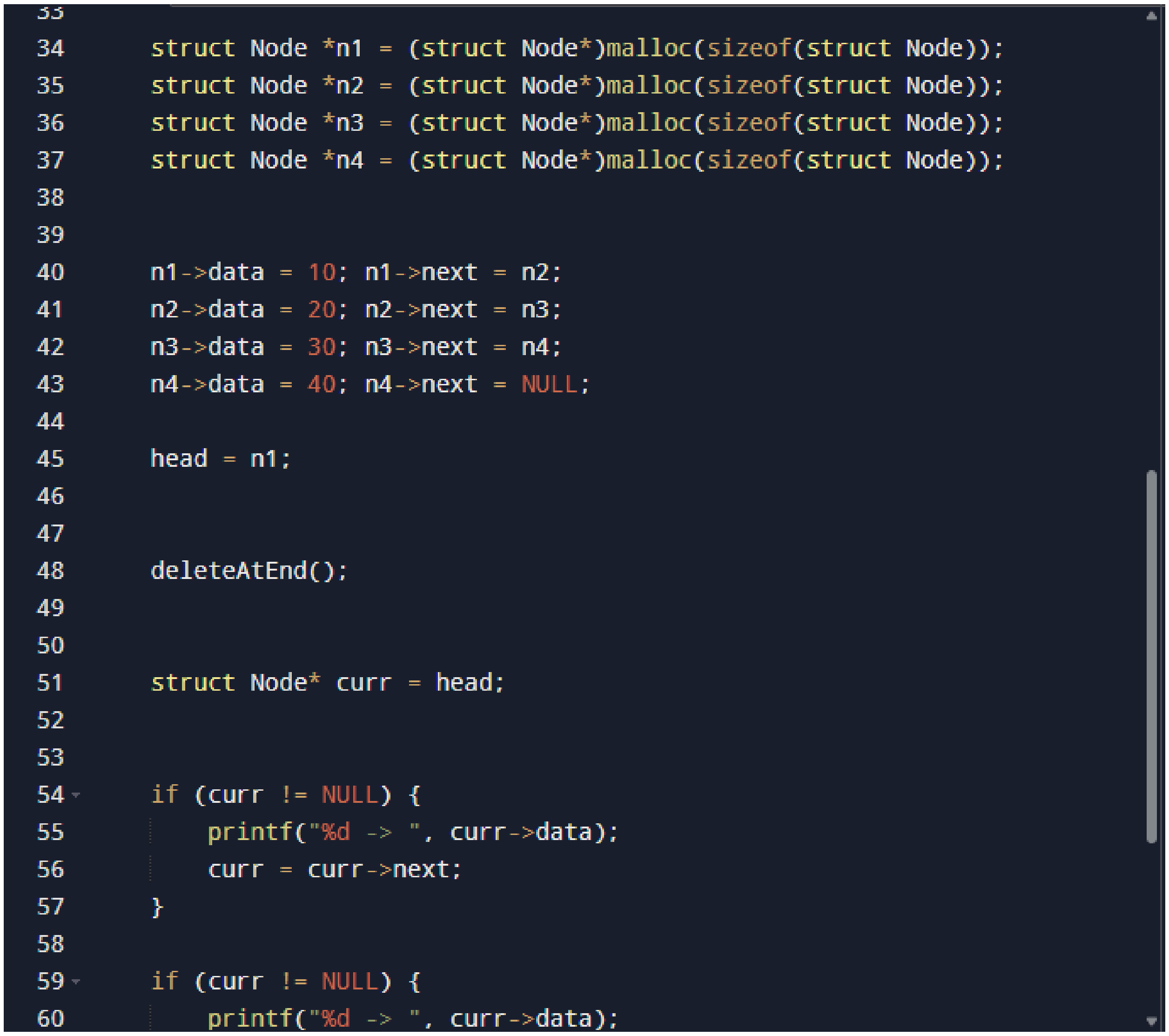
Input:

1RUA25BCA0096 Sanjana Pal



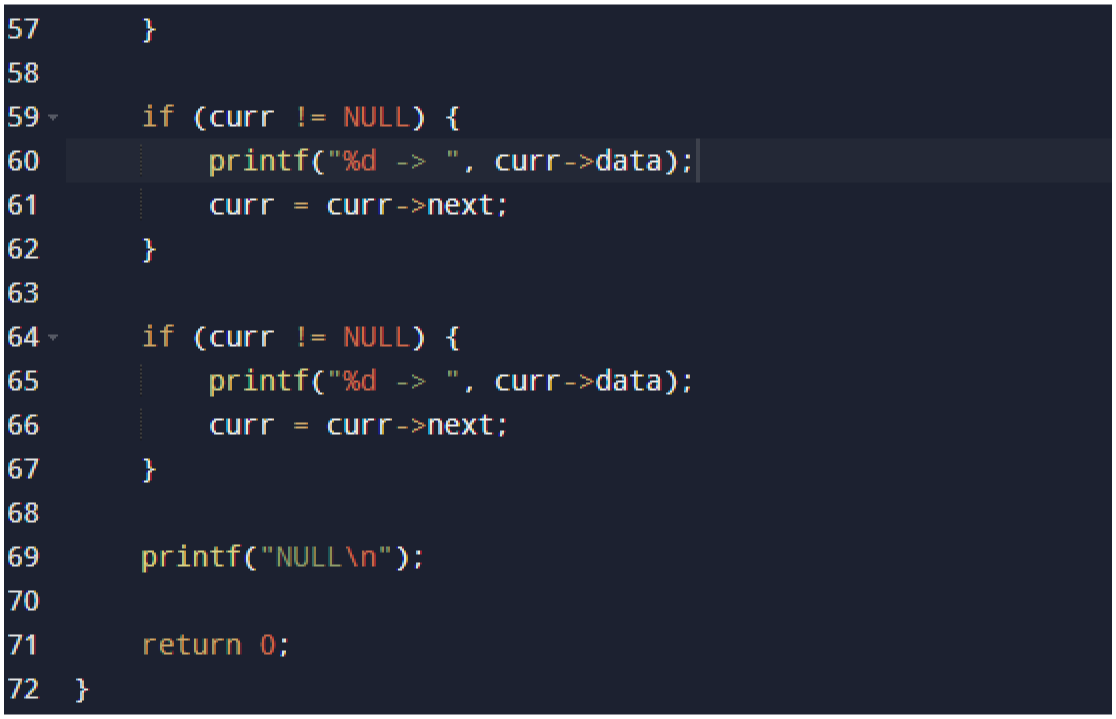
```c
#include <stdio.h>
#include <stdlib.h>

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


struct Node* head = NULL;

void deleteAtEnd() {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    if (head->next == NULL) {
        free(head);
        head = NULL;
        return;
    }

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

```

1RUA25BCA0096 Sanjana Pal



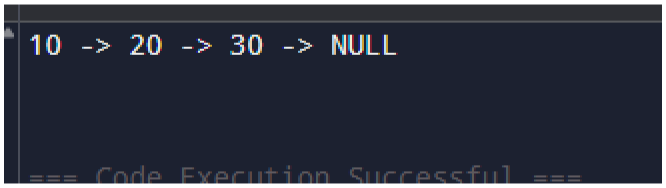
```
    struct Node *n1 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n2 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n3 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n4 = (struct Node*)malloc(sizeof(struct Node));


    n1->data = 10; n1->next = n2;
    n2->data = 20; n2->next = n3;
    n3->data = 30; n3->next = n4;
    n4->data = 40; n4->next = NULL;

    head = n1;


    deleteAtEnd();


    struct Node* curr = head;


    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }

    if (curr != NULL) {
        printf("%d -> ", curr->data);
```

1RUA25BCA0096 Sanjana Pal



```
    }

    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }

    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }

    printf("NULL\n");

    return 0;
}
```

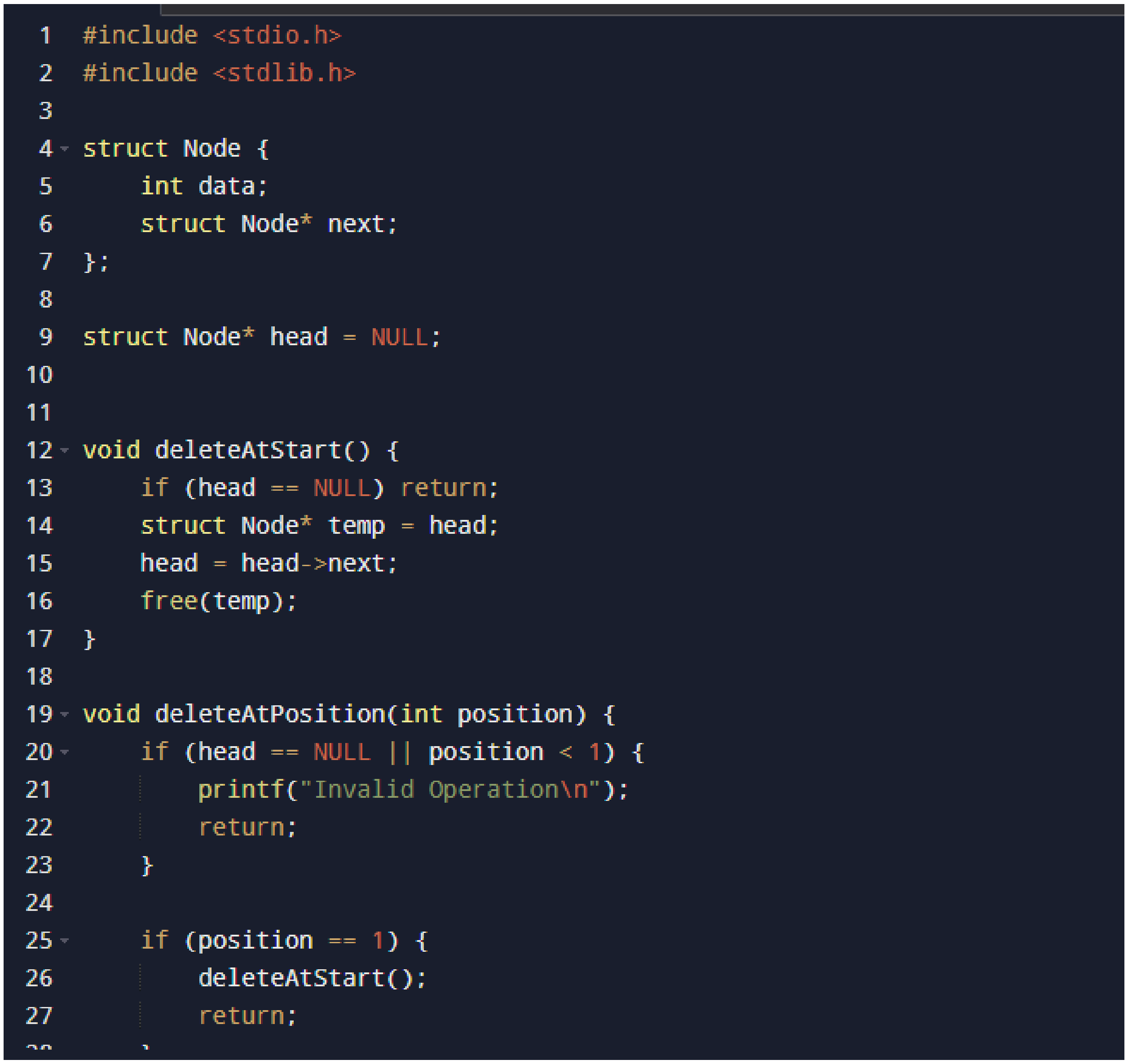
Output:



```
10 -> 20 -> 30 -> NULL


=== Code Execution Successful ===
```

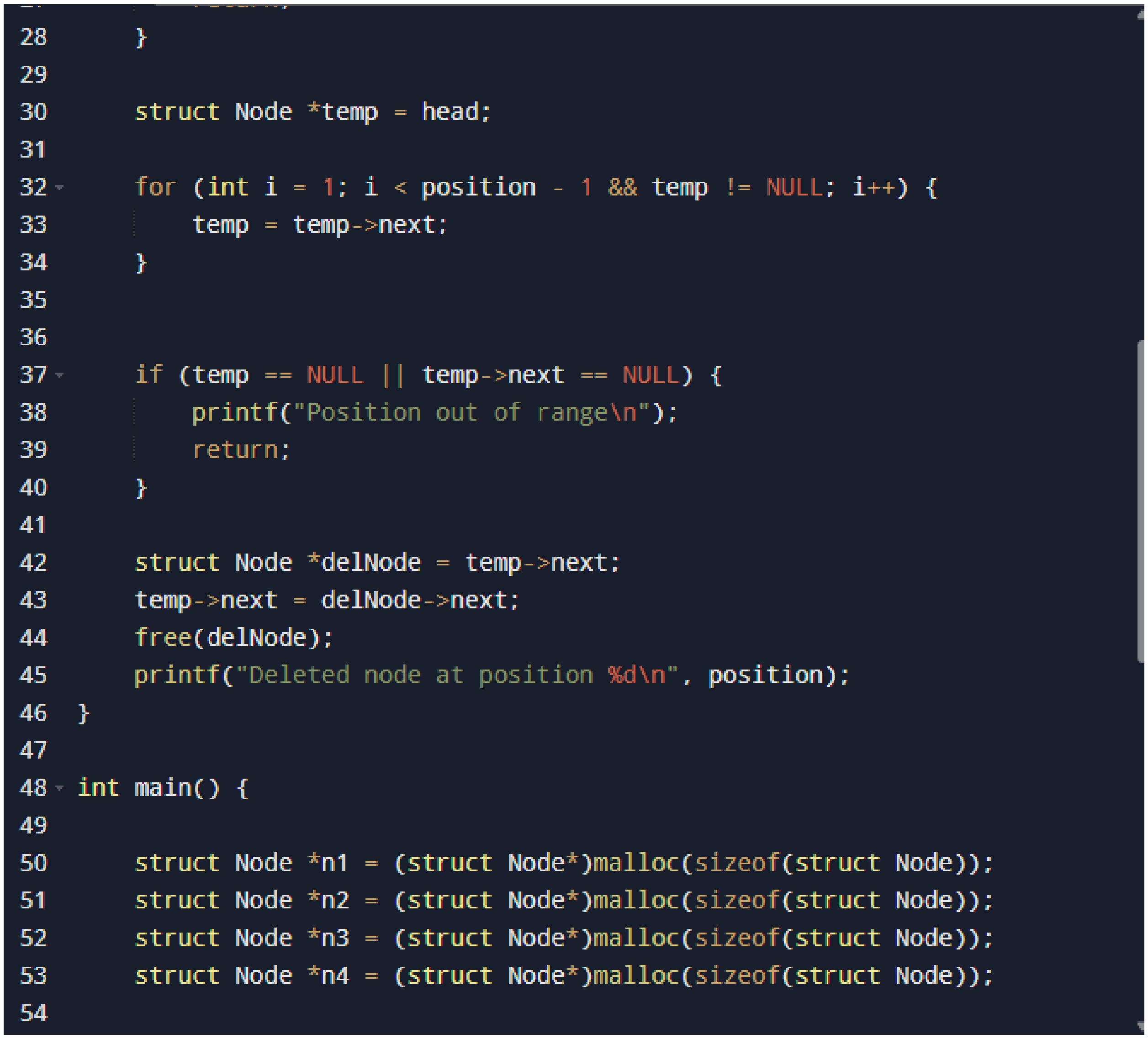
Deletion at position
Input:

1RUA25BCA0096 Sanjana Pal



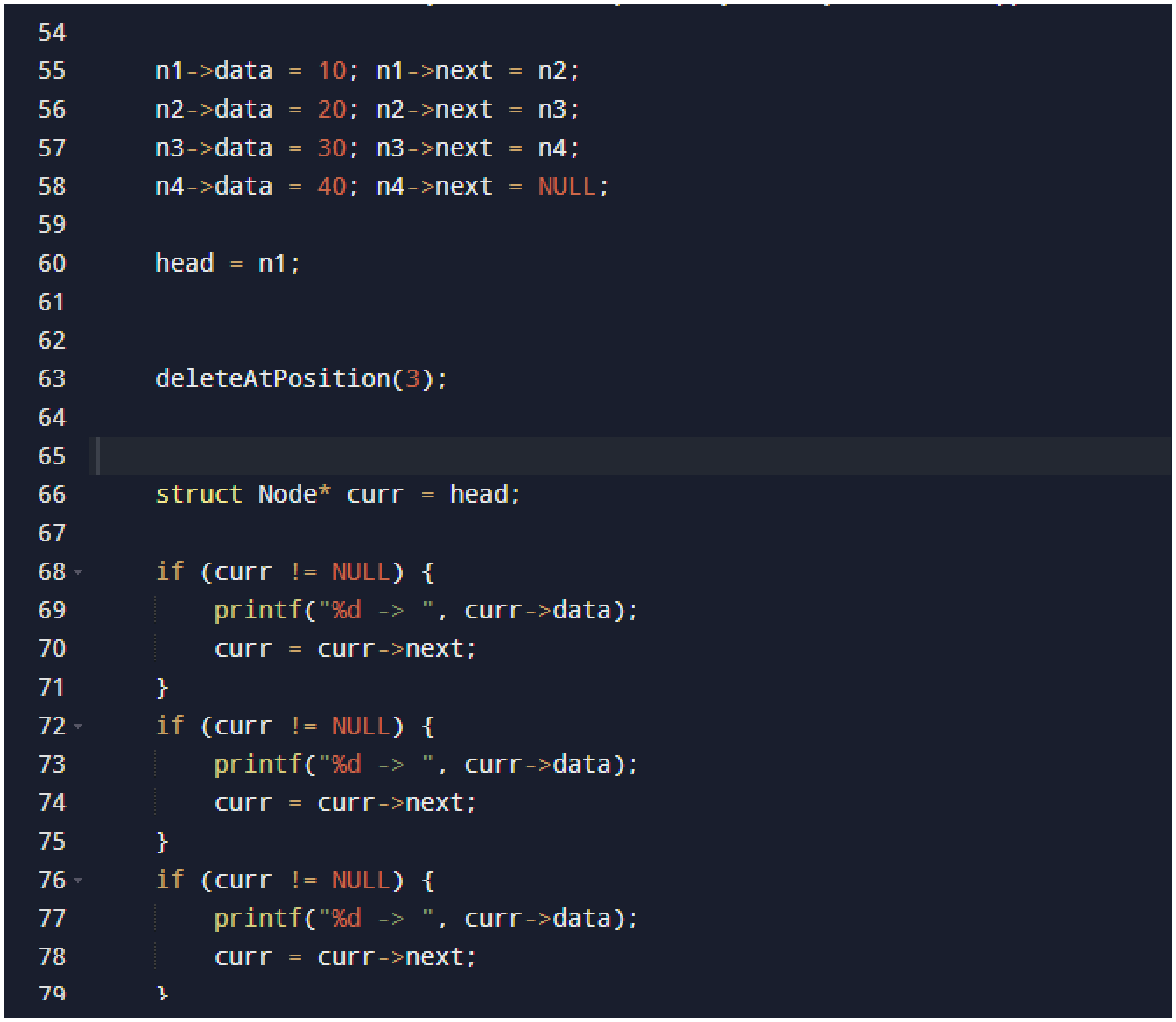
```c
#include <stdio.h>
#include <stdlib.h>

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

struct Node* head = NULL;


void deleteAtStart() {
    if (head == NULL) return;
    struct Node* temp = head;
    head = head->next;
    free(temp);
}

void deleteAtPosition(int position) {
    if (head == NULL || position < 1) {
        printf("Invalid Operation\n");
        return;
    }

    if (position == 1) {
        deleteAtStart();
        return;
```

1RUA25BCA0096 Sanjana Pal



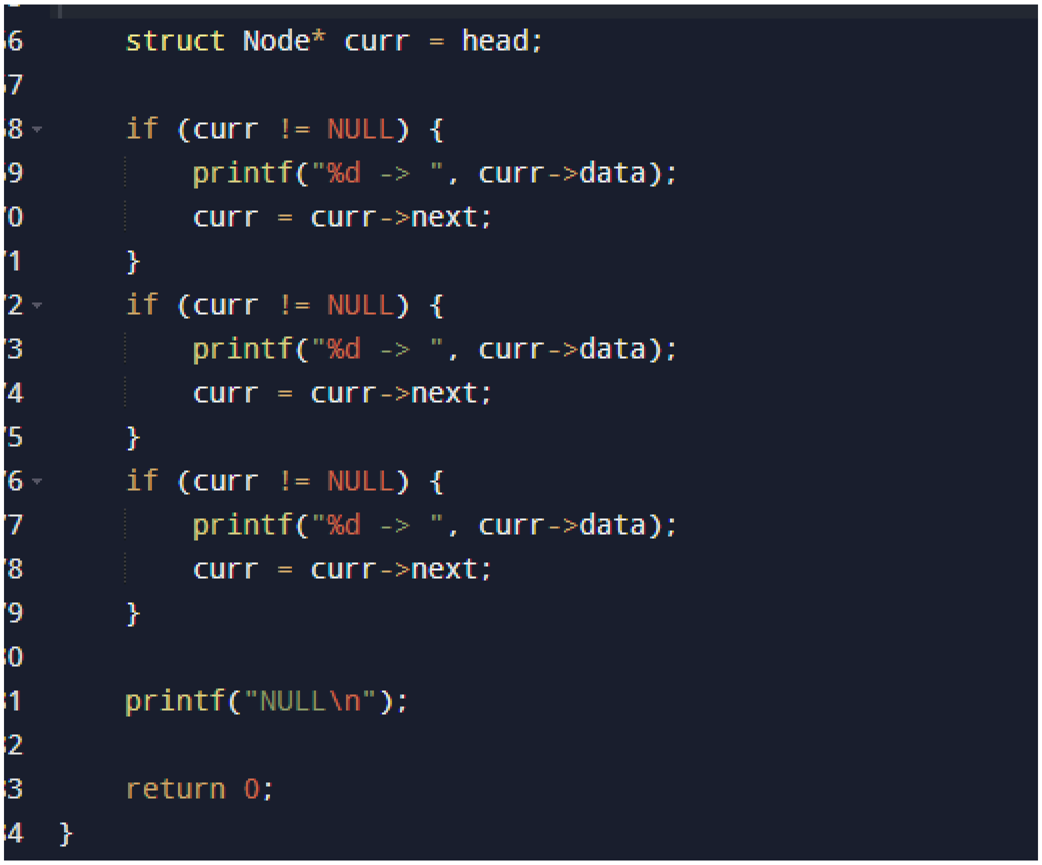
```
    }

    struct Node *temp = head;

    for (int i = 1; i < position - 1 && temp != NULL; i++) {
        temp = temp->next;
    }


    if (temp == NULL || temp->next == NULL) {
        printf("Position out of range\n");
        return;
    }

    struct Node *delNode = temp->next;
    temp->next = delNode->next;
    free(delNode);
    printf("Deleted node at position %d\n", position);
}

int main() {

    struct Node *n1 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n2 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n3 = (struct Node*)malloc(sizeof(struct Node));
    struct Node *n4 = (struct Node*)malloc(sizeof(struct Node));

```

1RUA25BCA0096 Sanjana Pal



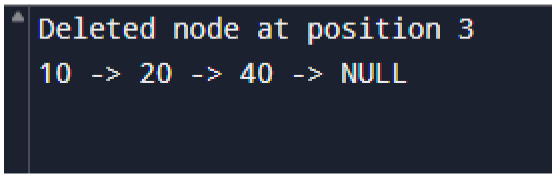
```c

    n1->data = 10; n1->next = n2;
    n2->data = 20; n2->next = n3;
    n3->data = 30; n3->next = n4;
    n4->data = 40; n4->next = NULL;

    head = n1;


    deleteAtPosition(3);


    struct Node* curr = head;

    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }
    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }
    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }
```

1RUA25BCA0096 Sanjana Pal



```
    struct Node* curr = head;

    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }
    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }
    if (curr != NULL) {
        printf("%d -> ", curr->data);
        curr = curr->next;
    }

    printf("NULL\n");

    return 0;
}
```

Output:



```
Deleted node at position 3
10 -> 20 -> 40 -> NULL
```