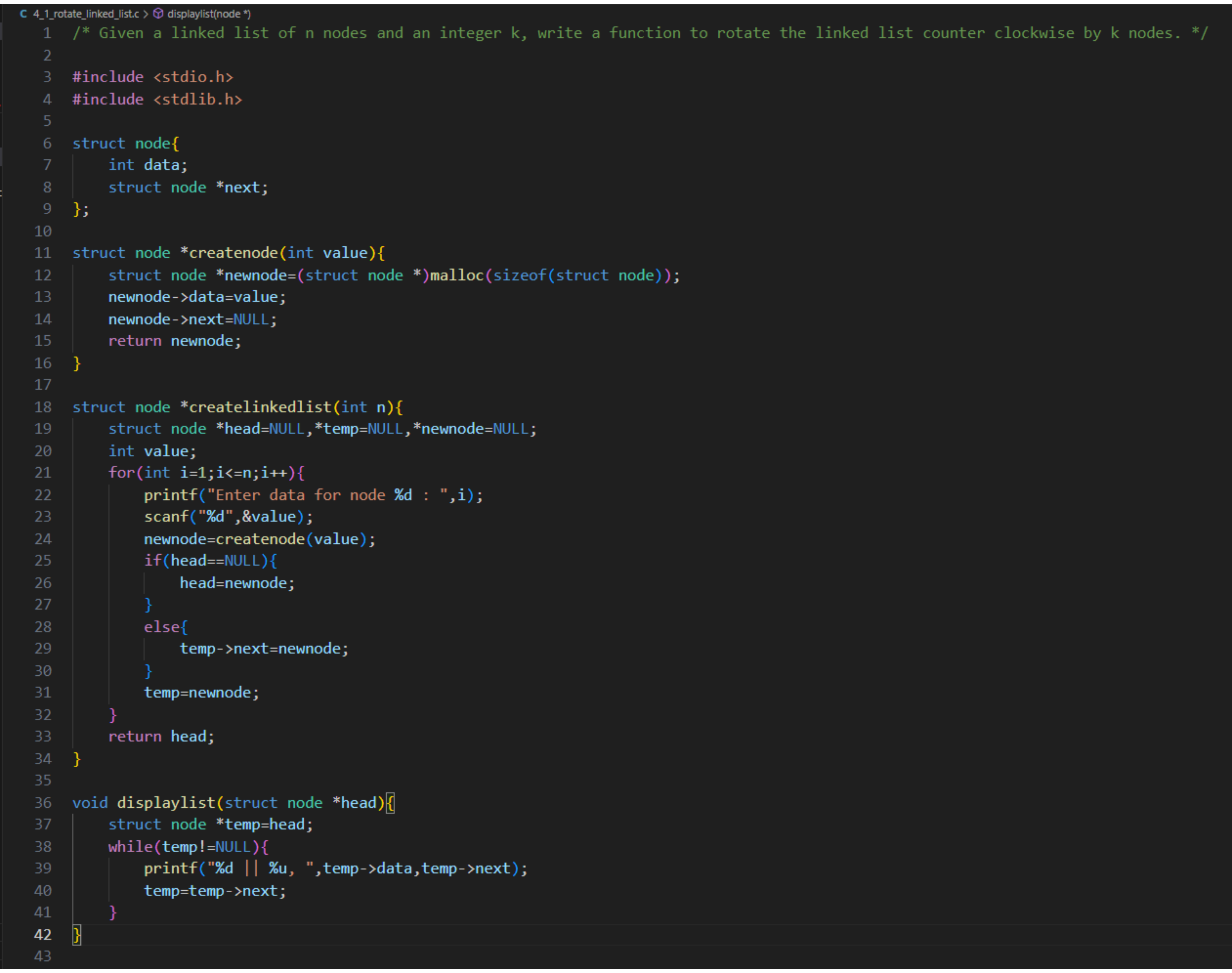
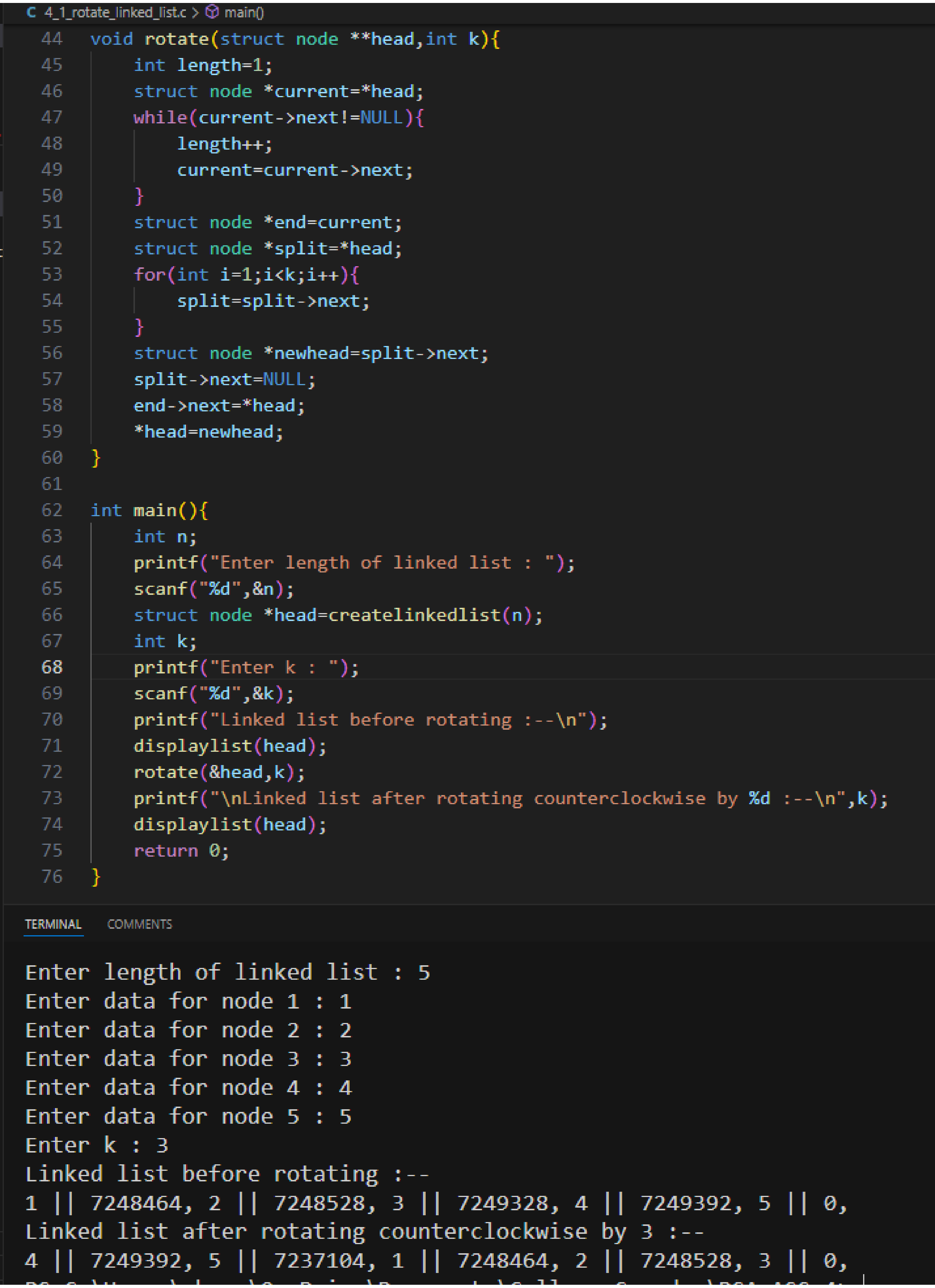
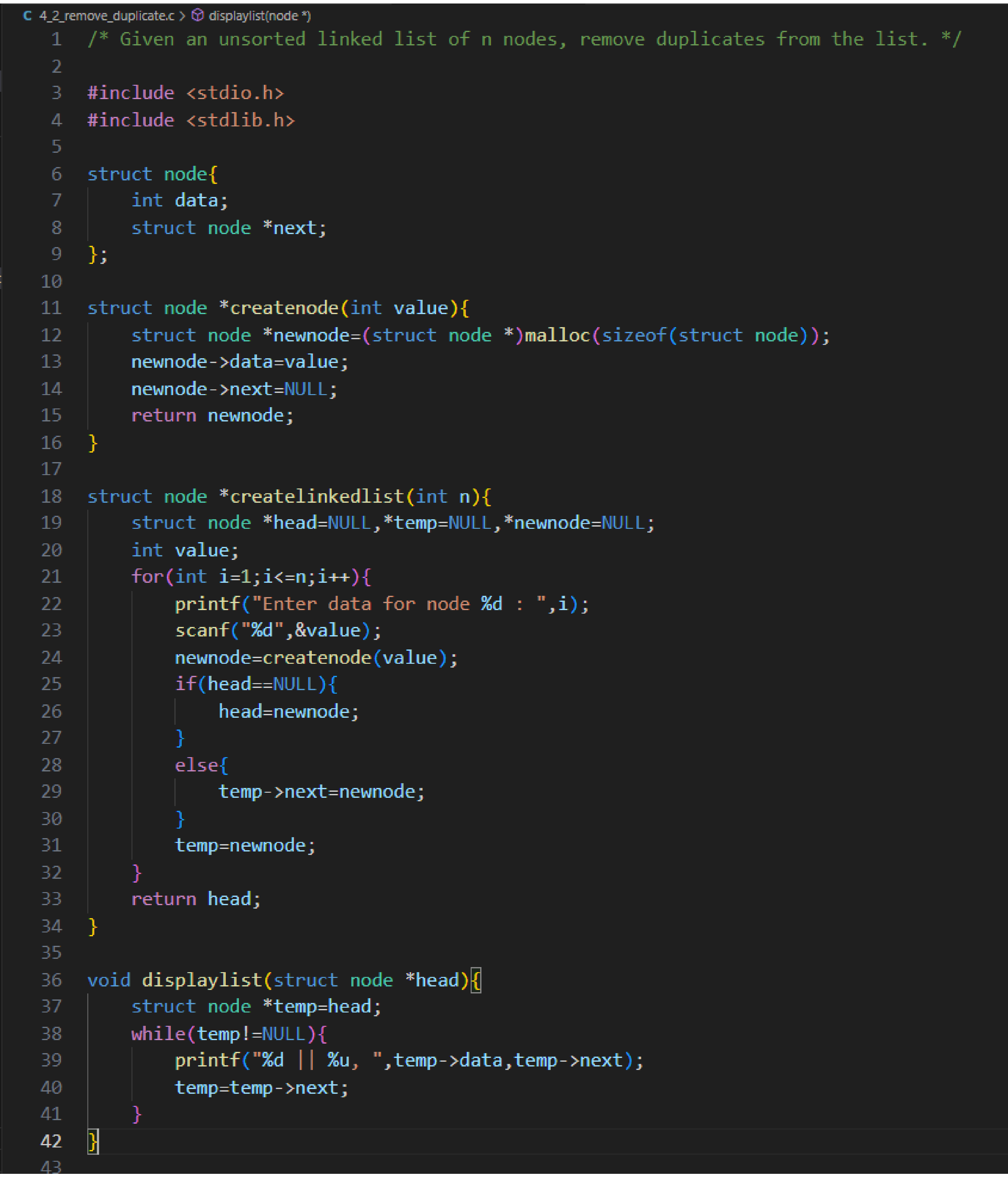


```c
/* Given a linked list of n nodes and an integer k, write a function to rotate the linked list counter clockwise by k nodes. */

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

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

struct node *createnode(int value){
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->next=NULL;
    return newnode;
}

struct node *createlinkedlist(int n){
    struct node *head=NULL,*temp=NULL,*newnode=NULL;
    int value;
    for(int i=1;i<=n;i++){
        printf("Enter data for node %d : ",i);
        scanf("%d",&value);
        newnode=createnode(value);
        if(head==NULL){
            head=newnode;
        }
        else{
            temp->next=newnode;
        }
        temp=newnode;
    }
    return head;
}

void displaylist(struct node *head){
    struct node *temp=head;
    while(temp!=NULL){
        printf("%d || %u, ",temp->data,temp->next);
        temp=temp->next;
    }
}

```

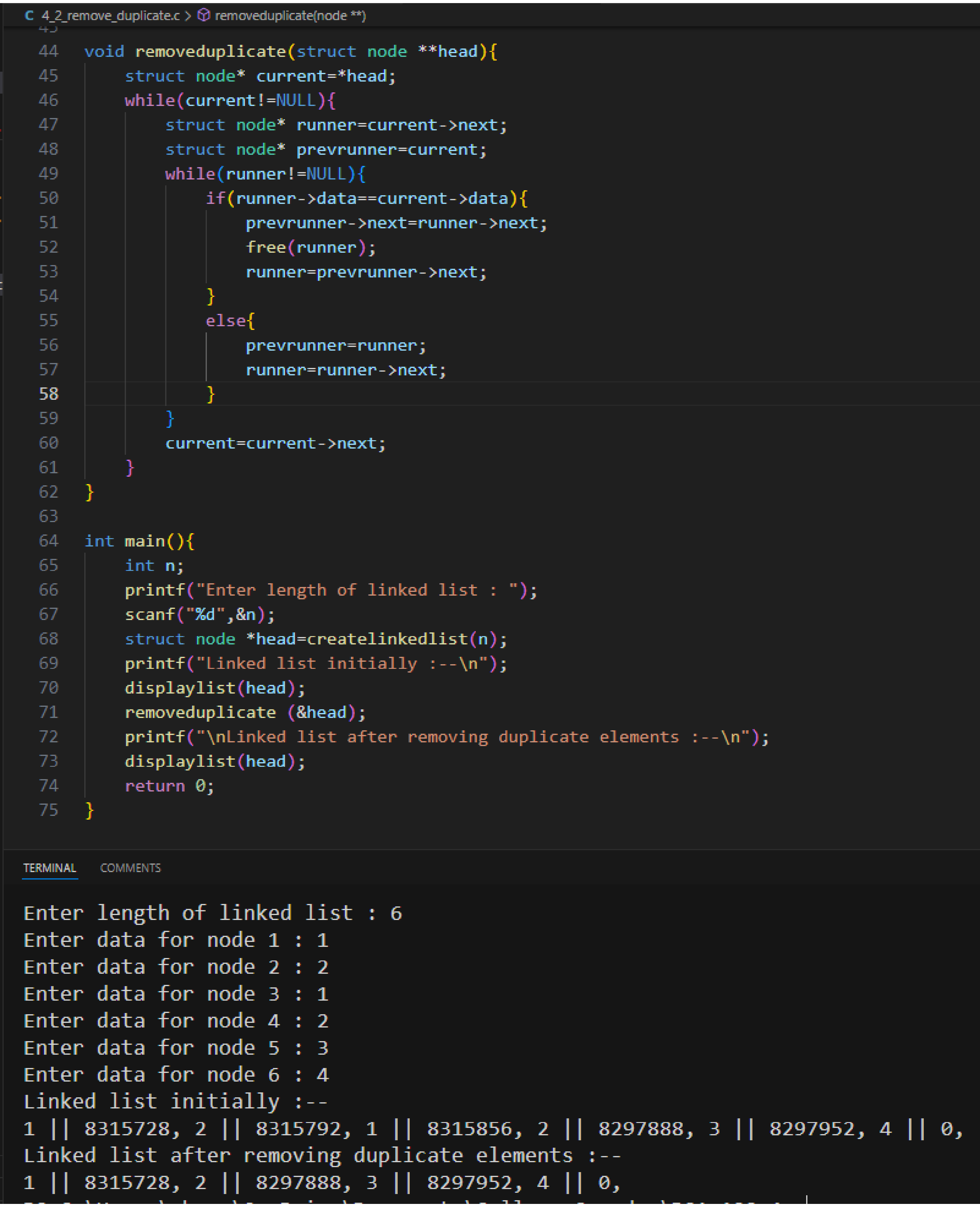


```c
void rotate(struct node **head,int k){
    int length=1;
    struct node *current=*head;
    while(current->next!=NULL){
        length++;
        current=current->next;
    }
    struct node *end=current;
    struct node *split=*head;
    for(int i=1;i<k;i++){
        split=split->next;
    }
    struct node *newhead=split->next;
    split->next=NULL;
    end->next=*head;
    *head=newhead;
}

int main(){
    int n;
    printf("Enter length of linked list : ");
    scanf("%d",&n);
    struct node *head=createlinkedlist(n);
    int k;
    printf("Enter k : ");
    scanf("%d",&k);
    printf("Linked list before rotating :--\n");
    displaylist(head);
    rotate(&head,k);
    printf("\nLinked list after rotating counterclockwise by %d :--\n",k);
    displaylist(head);
    return 0;
}
```

```
Enter length of linked list : 5
Enter data for node 1 : 1
Enter data for node 2 : 2
Enter data for node 3 : 3
Enter data for node 4 : 4
Enter data for node 5 : 5
Enter k : 3
Linked list before rotating :--
1 || 7248464, 2 || 7248528, 3 || 7249328, 4 || 7249392, 5 || 0,
Linked list after rotating counterclockwise by 3 :--
4 || 7249392, 5 || 7237104, 1 || 7248464, 2 || 7248528, 3 || 0,
```

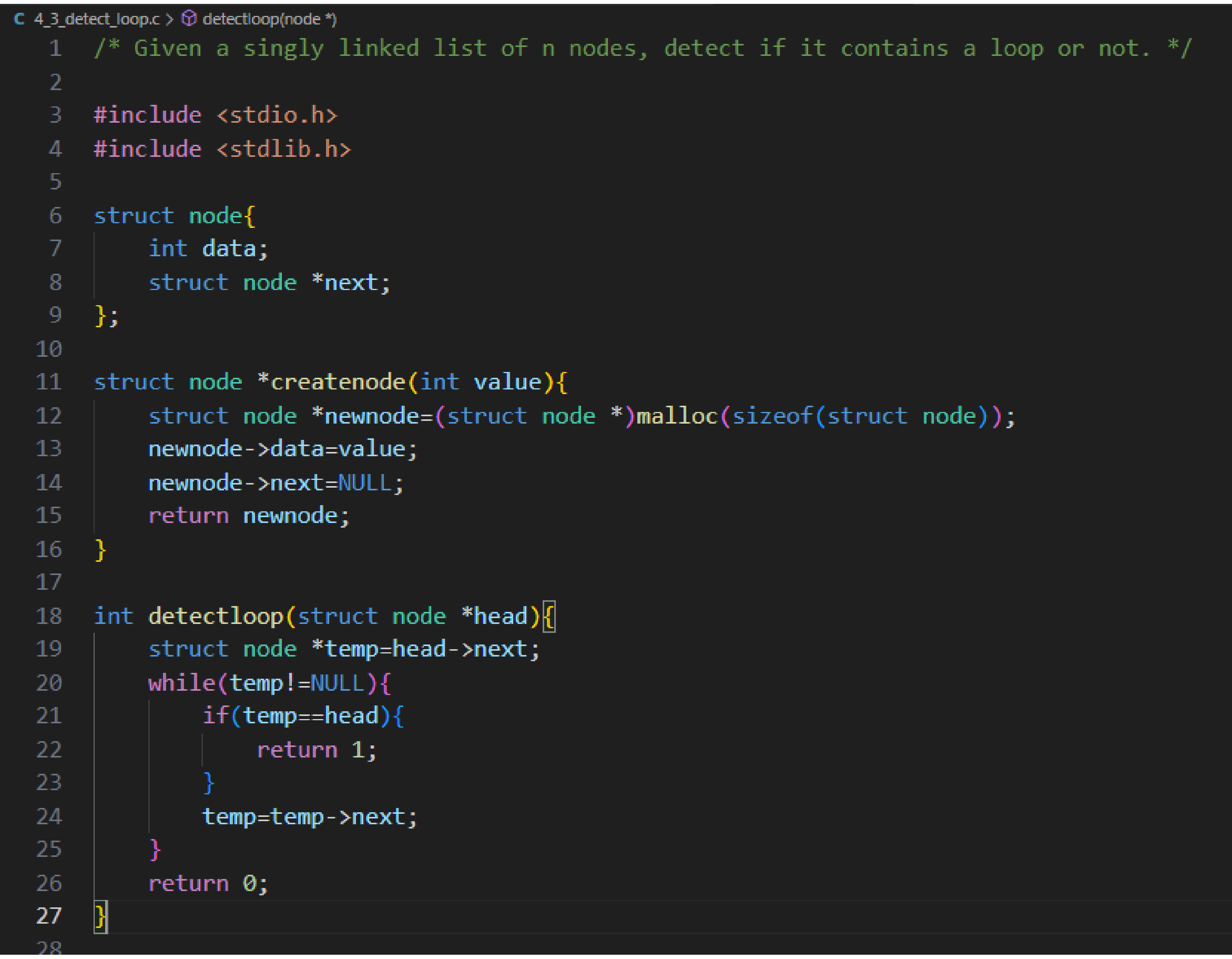


```c
/* Given an unsorted linked list of n nodes, remove duplicates from the list. */

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

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

struct node *createnode(int value){
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->next=NULL;
    return newnode;
}

struct node *createlinkedlist(int n){
    struct node *head=NULL,*temp=NULL,*newnode=NULL;
    int value;
    for(int i=1;i<=n;i++){
        printf("Enter data for node %d : ",i);
        scanf("%d",&value);
        newnode=createnode(value);
        if(head==NULL){
            head=newnode;
        }
        else{
            temp->next=newnode;
        }
        temp=newnode;
    }
    return head;
}

void displaylist(struct node *head){
    struct node *temp=head;
    while(temp!=NULL){
        printf("%d || %u, ",temp->data,temp->next);
        temp=temp->next;
    }
}

```

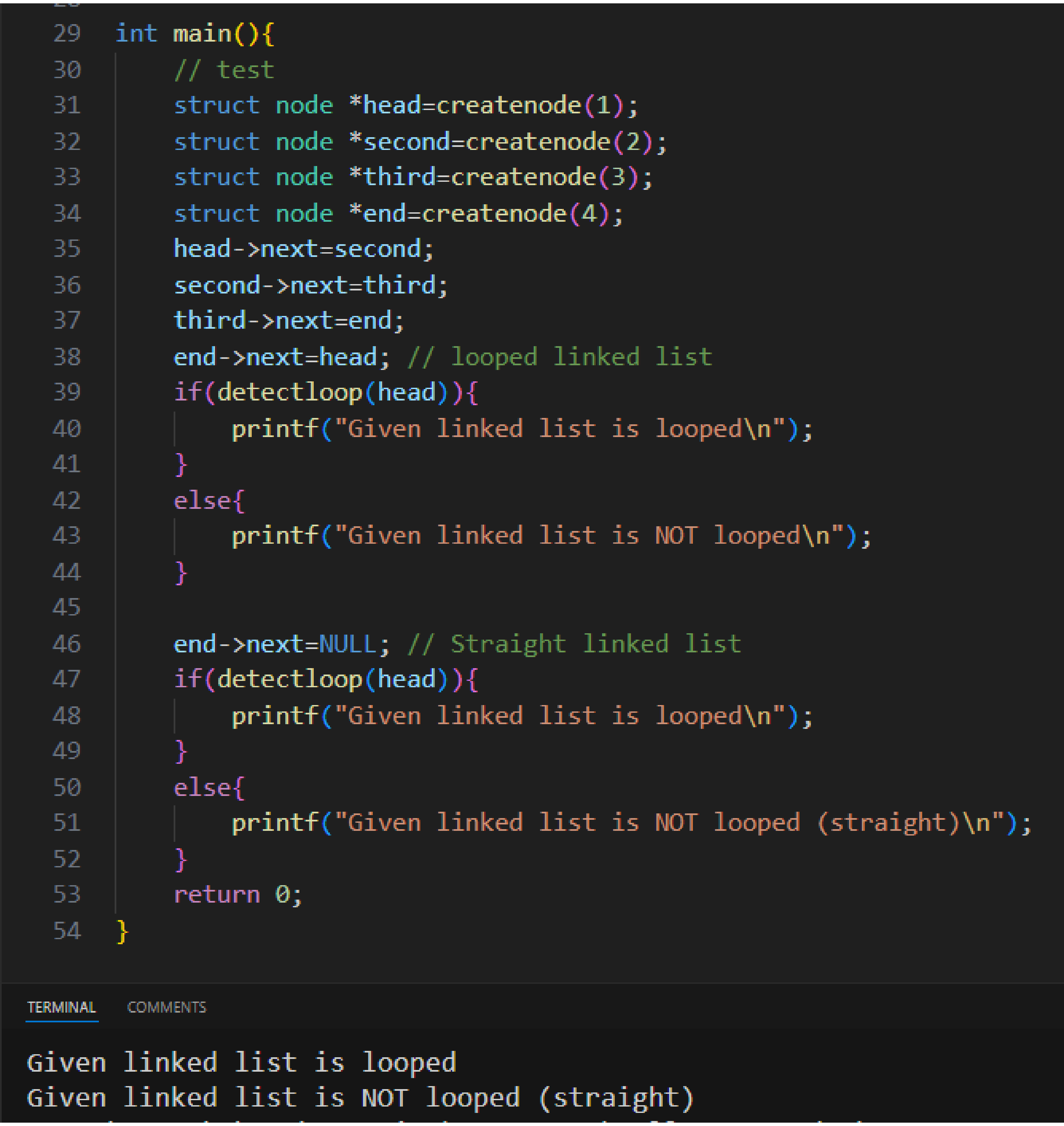


```c
void removeduplicate(struct node **head){
    struct node* current=*head;
    while(current!=NULL){
        struct node* runner=current->next;
        struct node* prevrunner=current;
        while(runner!=NULL){
            if(runner->data==current->data){
                prevrunner->next=runner->next;
                free(runner);
                runner=prevrunner->next;
            }
            else{
                prevrunner=runner;
                runner=runner->next;
            }
        }
        current=current->next;
    }
}

int main(){
    int n;
    printf("Enter length of linked list : ");
    scanf("%d",&n);
    struct node *head=createlinkedlist(n);
    printf("Linked list initially :--\n");
    displaylist(head);
    removeduplicate (&head);
    printf("\nLinked list after removing duplicate elements :--\n");
    displaylist(head);
    return 0;
}
```

```
Enter length of linked list : 6
Enter data for node 1 : 1
Enter data for node 2 : 2
Enter data for node 3 : 1
Enter data for node 4 : 2
Enter data for node 5 : 3
Enter data for node 6 : 4
Linked list initially :--
1 || 8315728, 2 || 8315792, 1 || 8315856, 2 || 8297888, 3 || 8297952, 4 || 0,
Linked list after removing duplicate elements :--
1 || 8315728, 2 || 8297888, 3 || 8297952, 4 || 0,
```

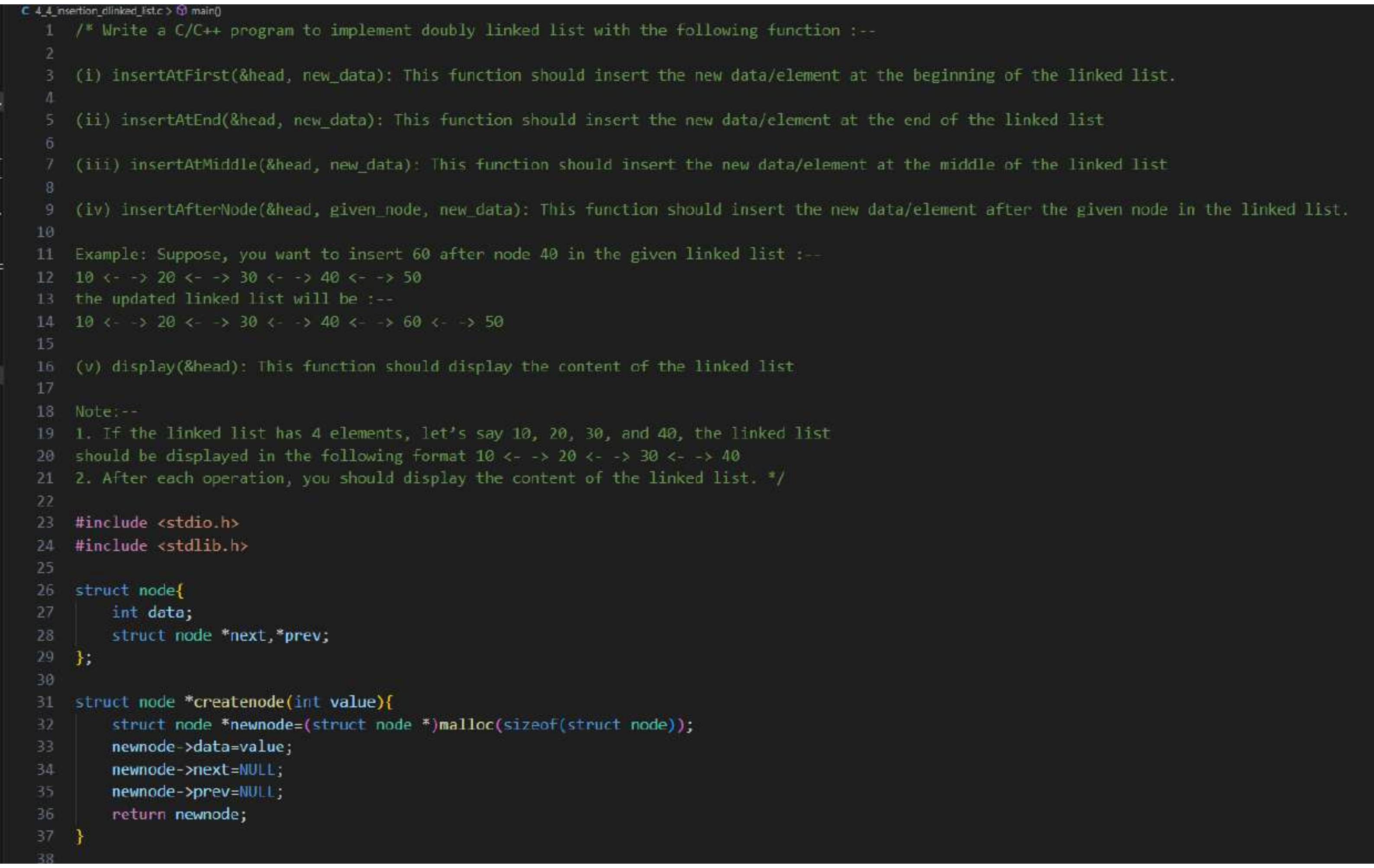

4_3_detect_loop.c > detectloop(node *)

```c
/* Given a singly linked list of n nodes, detect if it contains a loop or not. */

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

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

struct node *createnode(int value){
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->next=NULL;
    return newnode;
}

int detectloop(struct node *head){
    struct node *temp=head->next;
    while(temp!=NULL){
        if(temp==head){
            return 1;
        }
        temp=temp->next;
    }
    return 0;
}
```

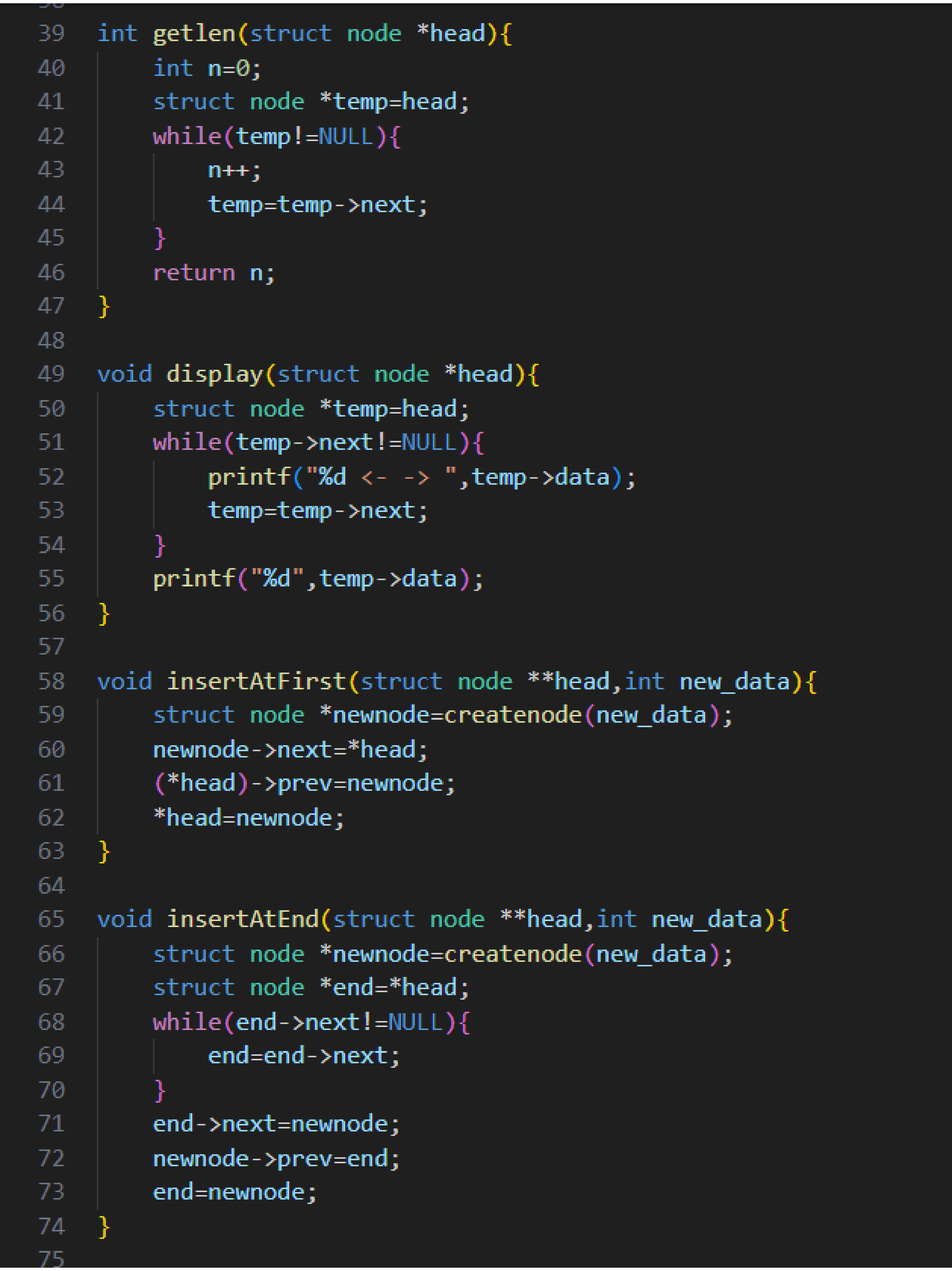


```c
int main(){
    // test
    struct node *head=createnode(1);
    struct node *second=createnode(2);
    struct node *third=createnode(3);
    struct node *end=createnode(4);
    head->next=second;
    second->next=third;
    third->next=end;
    end->next=head; // looped linked list
    if(detectloop(head)){
        printf("Given linked list is looped\n");
    }
    else{
        printf("Given linked list is NOT looped\n");
    }

    end->next=NULL; // Straight linked list
    if(detectloop(head)){
        printf("Given linked list is looped\n");
    }
    else{
        printf("Given linked list is NOT looped (straight)\n");
    }
    return 0;
}
```

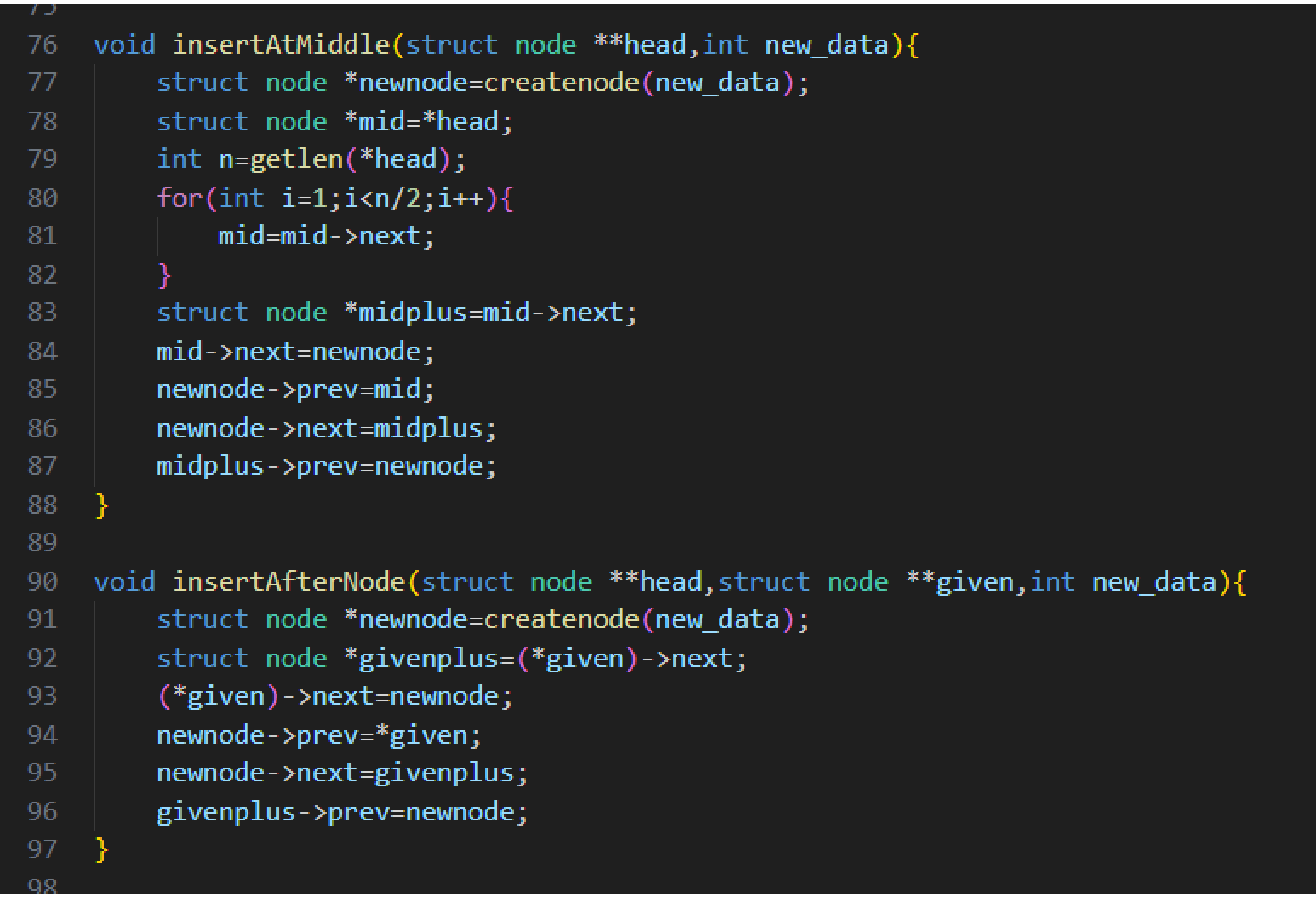
```
Given linked list is looped
Given linked list is NOT looped (straight)
```



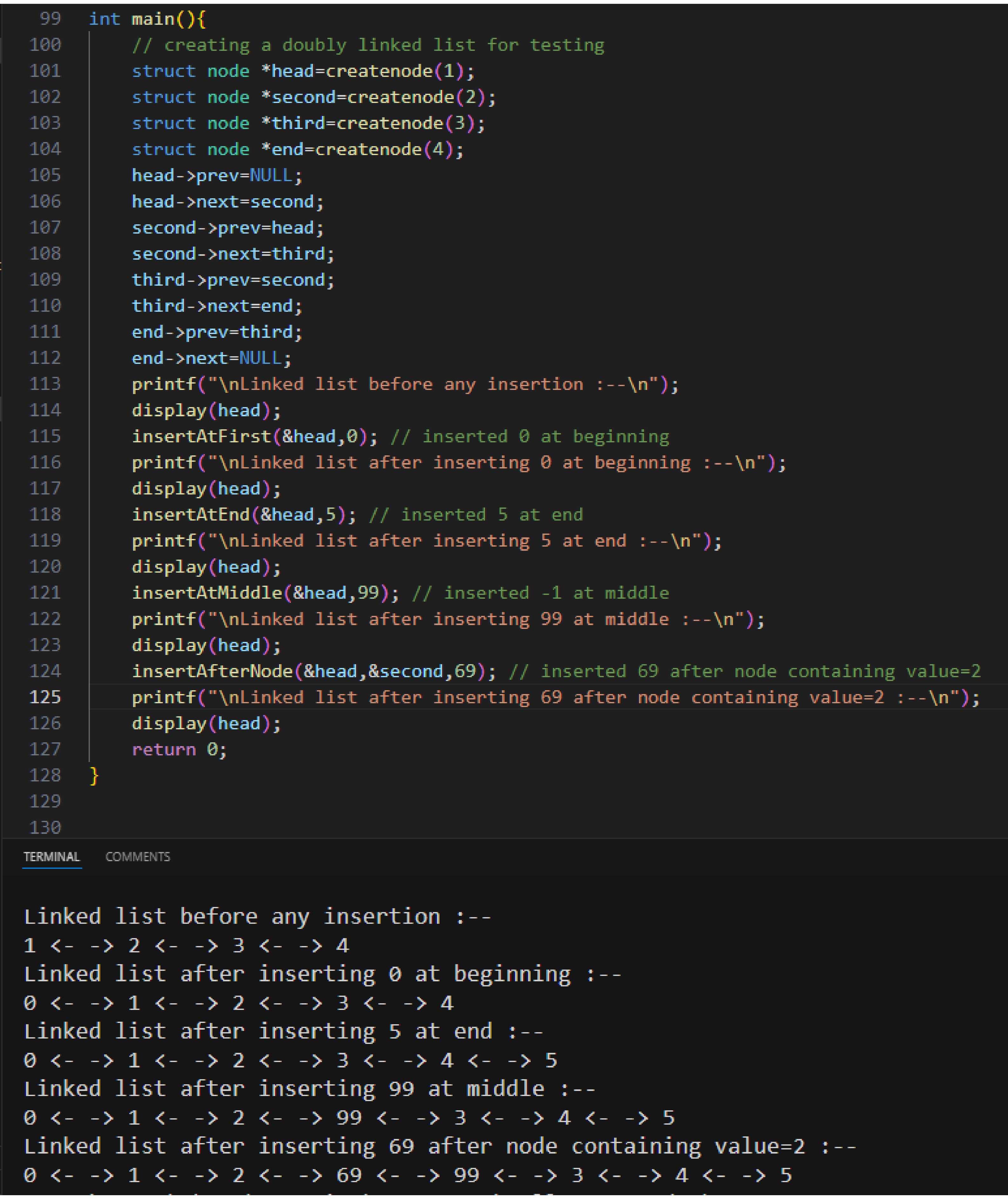
```c
/* Write a C/C++ program to implement doubly linked list with the following function :--

(i) insertAtFirst(&head, new_data): This function should insert the new data/element at the beginning of the linked list.

(ii) insertAtEnd(&head, new_data): This function should insert the new data/element at the end of the linked list

(iii) insertAtMiddle(&head, new_data): This function should insert the new data/element at the middle of the linked list

(iv) insertAfterNode(&head, given_node, new_data): This function should insert the new data/element after the given node in the linked list.

Example: Suppose, you want to insert 60 after node 40 in the given linked list :--
10 <- -> 20 <- -> 30 <- -> 40 <- -> 50
the updated linked list will be :--
10 <- -> 20 <- -> 30 <- -> 40 <- -> 60 <- -> 50

(v) display(&head): This function should display the content of the linked list

Note:--
1. If the linked list has 4 elements, let’s say 10, 20, 30, and 40, the linked list
should be displayed in the following format 10 <- -> 20 <- -> 30 <- -> 40
2. After each operation, you should display the content of the linked list. */

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

struct node{
    int data;
    struct node *next,*prev;
};

struct node *createnode(int value){
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=value;
    newnode->next=NULL;
    newnode->prev=NULL;
    return newnode;
}

```



```
int getlen(struct node *head){
    int n=0;
    struct node *temp=head;
    while(temp!=NULL){
        n++;
        temp=temp->next;
    }
    return n;
}

void display(struct node *head){
    struct node *temp=head;
    while(temp->next!=NULL){
        printf("%d <- -> ",temp->data);
        temp=temp->next;
    }
    printf("%d",temp->data);
}

void insertAtFirst(struct node **head,int new_data){
    struct node *newnode=createnode(new_data);
    newnode->next=*head;
    (*head)->prev=newnode;
    *head=newnode;
}

void insertAtEnd(struct node **head,int new_data){
    struct node *newnode=createnode(new_data);
    struct node *end=*head;
    while(end->next!=NULL){
        end=end->next;
    }
    end->next=newnode;
    newnode->prev=end;
    end=newnode;
}
```



```c
void insertAtMiddle(struct node **head,int new_data){
    struct node *newnode=createnode(new_data);
    struct node *mid=*head;
    int n=getlen(*head);
    for(int i=1;i<n/2;i++){
        mid=mid->next;
    }
    struct node *midplus=mid->next;
    mid->next=newnode;
    newnode->prev=mid;
    newnode->next=midplus;
    midplus->prev=newnode;
}

void insertAfterNode(struct node **head,struct node **given,int new_data){
    struct node *newnode=createnode(new_data);
    struct node *givenplus=(*given)->next;
    (*given)->next=newnode;
    newnode->prev=*given;
    newnode->next=givenplus;
    givenplus->prev=newnode;
}
```



```c
int main(){
    // creating a doubly linked list for testing
    struct node *head=createnode(1);
    struct node *second=createnode(2);
    struct node *third=createnode(3);
    struct node *end=createnode(4);
    head->prev=NULL;
    head->next=second;
    second->prev=head;
    second->next=third;
    third->prev=second;
    third->next=end;
    end->prev=third;
    end->next=NULL;
    printf("\nLinked list before any insertion :--\n");
    display(head);
    insertAtFirst(&head,0); // inserted 0 at beginning
    printf("\nLinked list after inserting 0 at beginning :--\n");
    display(head);
    insertAtEnd(&head,5); // inserted 5 at end
    printf("\nLinked list after inserting 5 at end :--\n");
    display(head);
    insertAtMiddle(&head,99); // inserted -1 at middle
    printf("\nLinked list after inserting 99 at middle :--\n");
    display(head);
    insertAfterNode(&head,&second,69); // inserted 69 after node containing value=2
    printf("\nLinked list after inserting 69 after node containing value=2 :--\n");
    display(head);
    return 0;
}
```

```
Linked list before any insertion :--
1 <- -> 2 <- -> 3 <- -> 4
Linked list after inserting 0 at beginning :--
0 <- -> 1 <- -> 2 <- -> 3 <- -> 4
Linked list after inserting 5 at end :--
0 <- -> 1 <- -> 2 <- -> 3 <- -> 4 <- -> 5
Linked list after inserting 99 at middle :--
0 <- -> 1 <- -> 2 <- -> 99 <- -> 3 <- -> 4 <- -> 5
Linked list after inserting 69 after node containing value=2 :--
0 <- -> 1 <- -> 2 <- -> 69 <- -> 99 <- -> 3 <- -> 4 <- -> 5
```