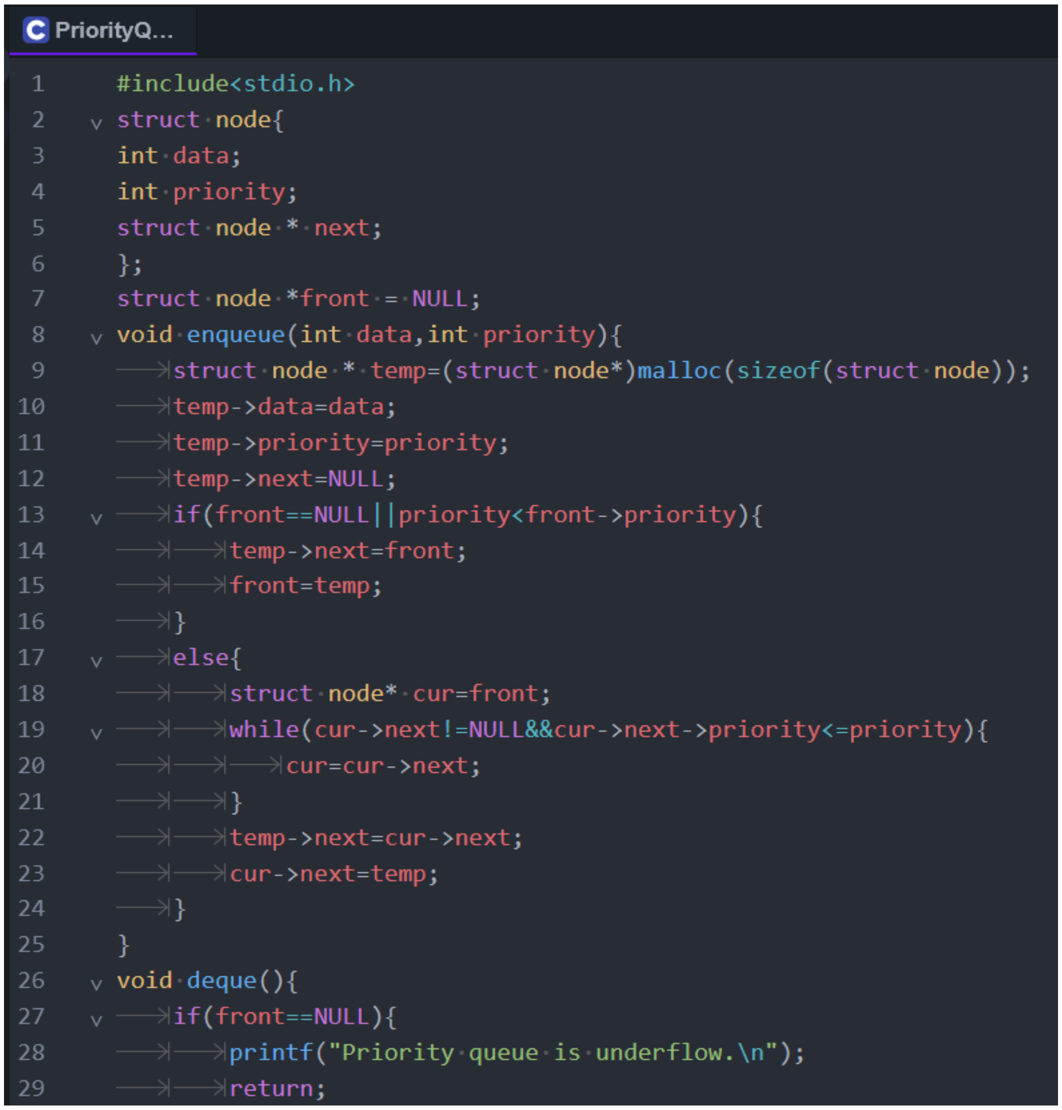
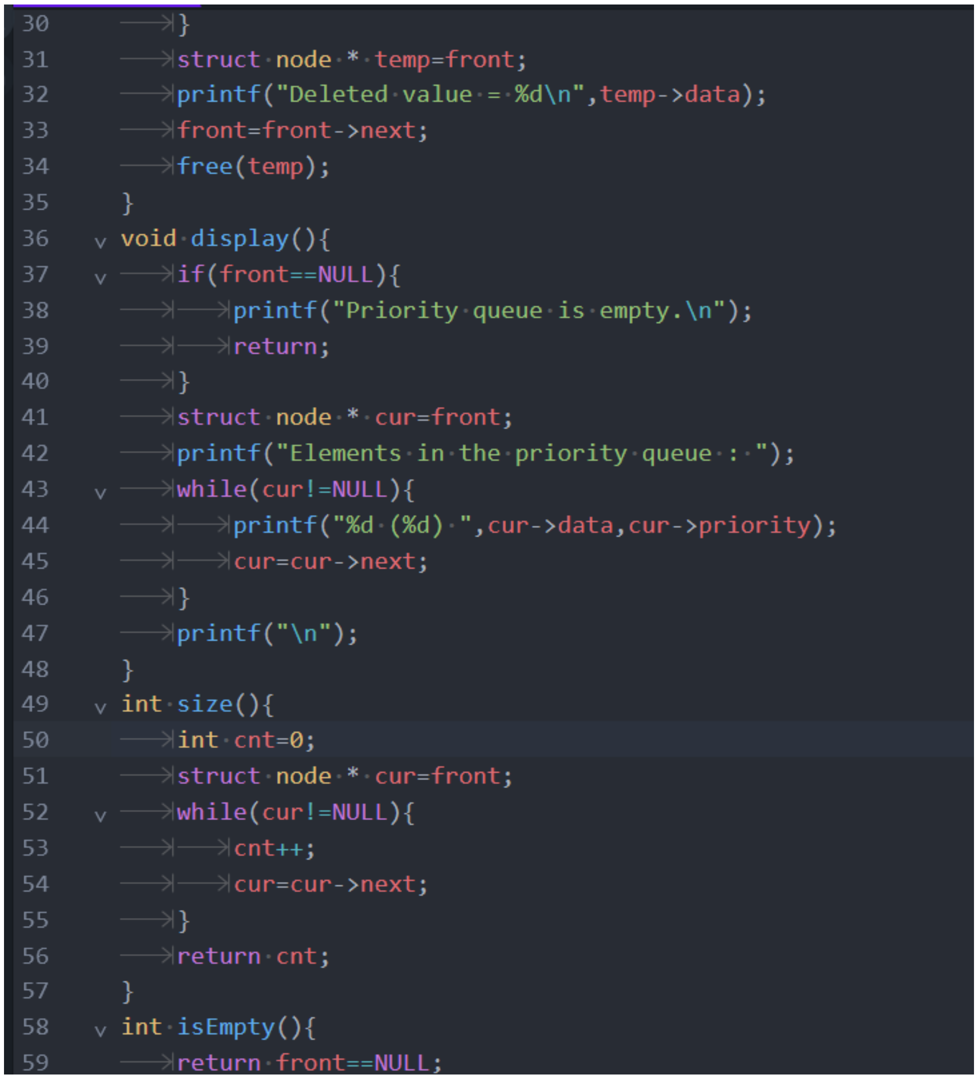
DS Lab 7 Solutions

7.1.1

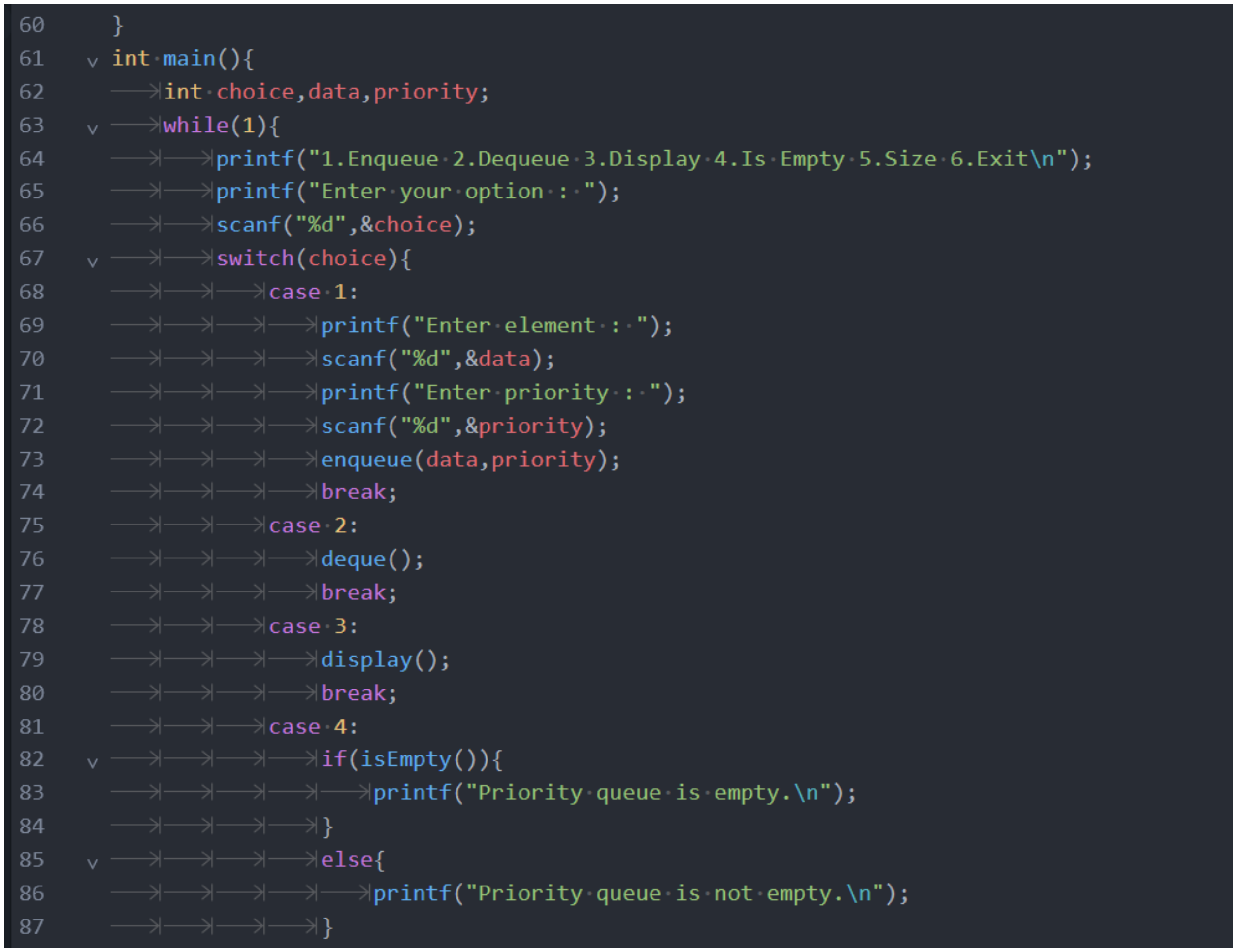


PriorityQ...

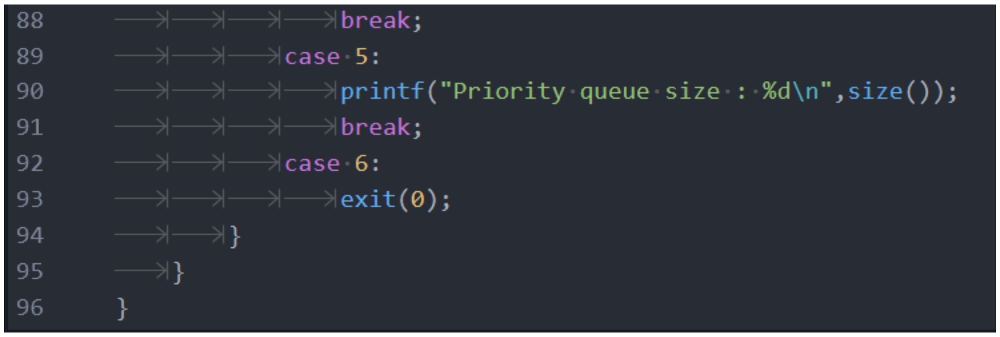
```c
#include<stdio.h>
struct node{
int data;
int priority;
struct node * next;
};
struct node *front = NULL;
void enqueue(int data,int priority){
	struct node * temp=(struct node*)malloc(sizeof(struct node));
	temp->data=data;
	temp->priority=priority;
	temp->next=NULL;
	if(front==NULL||priority<front->priority){
		temp->next=front;
		front=temp;
	}
	else{
		struct node* cur=front;
		while(cur->next!=NULL&&cur->next->priority<=priority){
			cur=cur->next;
		}
		temp->next=cur->next;
		cur->next=temp;
	}
}
void deque(){
	if(front==NULL){
		printf("Priority queue is underflow.\n");
		return;
```



```c
	}
	struct node * temp=front;
	printf("Deleted value = %d\n",temp->data);
	front=front->next;
	free(temp);
}
void display(){
	if(front==NULL){
		printf("Priority queue is empty.\n");
		return;
	}
	struct node * cur=front;
	printf("Elements in the priority queue : ");
	while(cur!=NULL){
		printf("%d (%d) ",cur->data,cur->priority);
		cur=cur->next;
	}
	printf("\n");
}
int size(){
	int cnt=0;
	struct node * cur=front;
	while(cur!=NULL){
		cnt++;
		cur=cur->next;
	}
	return cnt;
}
int isEmpty(){
	return front==NULL;
```

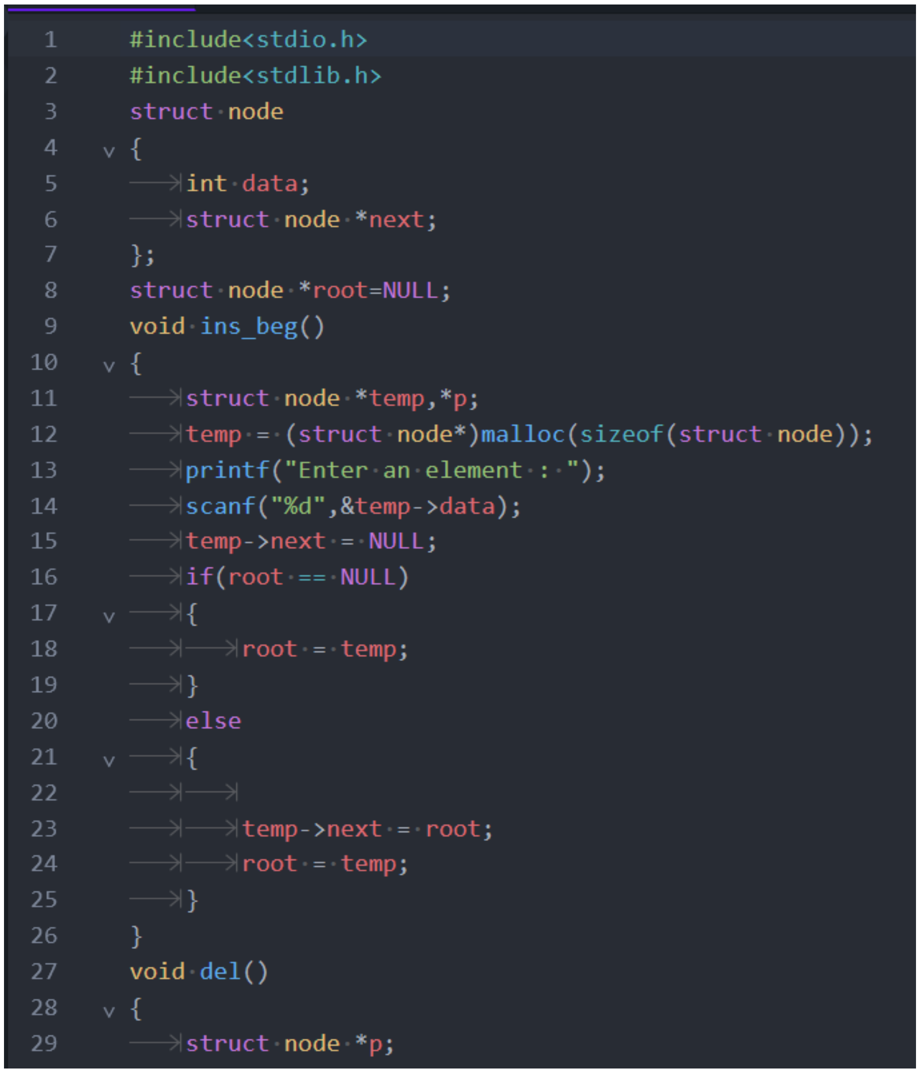


```c
}
int main(){
    int choice,data,priority;
    while(1){
        printf("1.Enqueue 2.Dequeue 3.Display 4.Is Empty 5.Size 6.Exit\n");
        printf("Enter your option : ");
        scanf("%d",&choice);
        switch(choice){
            case 1:
                printf("Enter element : ");
                scanf("%d",&data);
                printf("Enter priority : ");
                scanf("%d",&priority);
                enqueue(data,priority);
                break;
            case 2:
                deque();
                break;
            case 3:
                display();
                break;
            case 4:
                if(isEmpty()){
                    printf("Priority queue is empty.\n");
                }
                else{
                    printf("Priority queue is not empty.\n");
                }
```

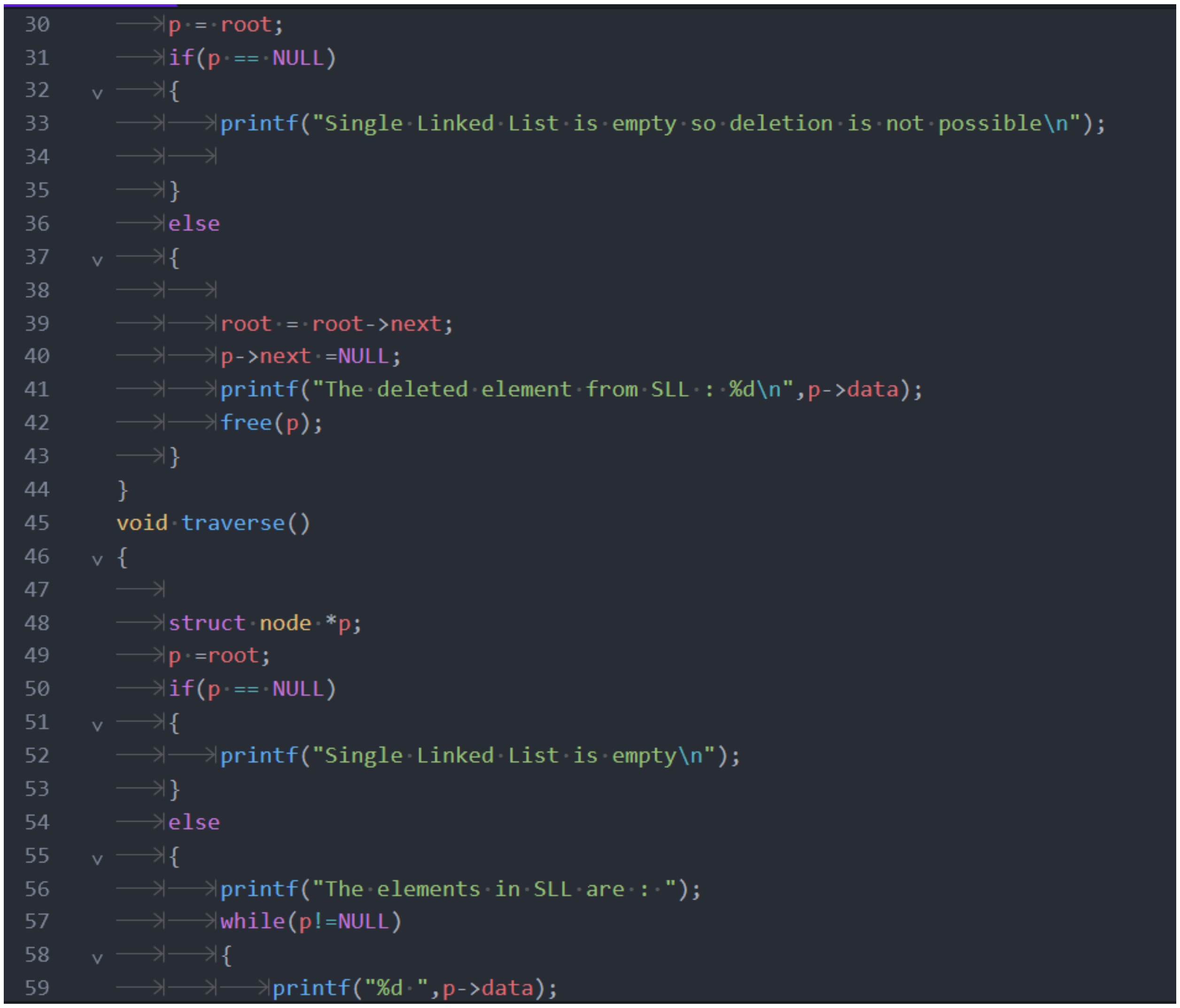


```c
                break;
            case 5:
                printf("Priority queue size : %d\n",size());
                break;
            case 6:
                exit(0);
        }
    }
}
```

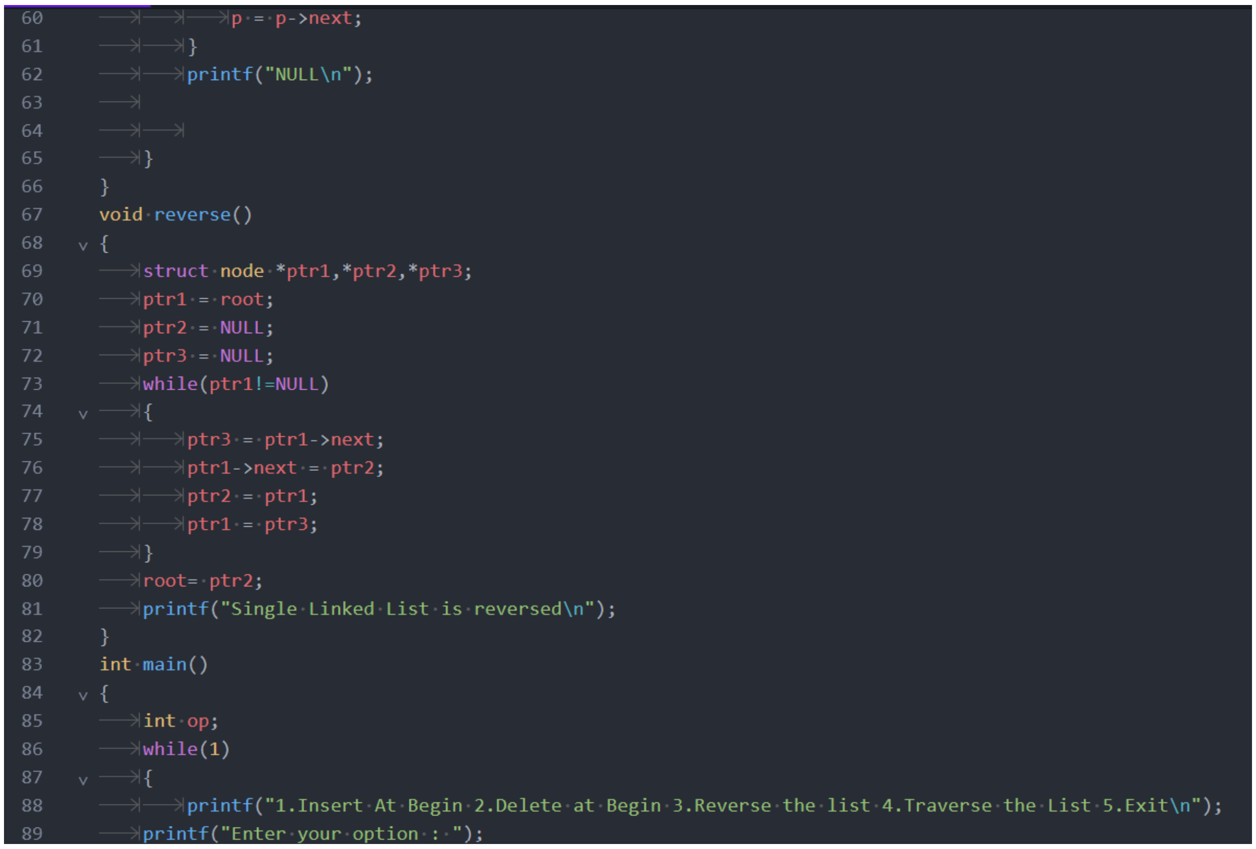
7.1.2



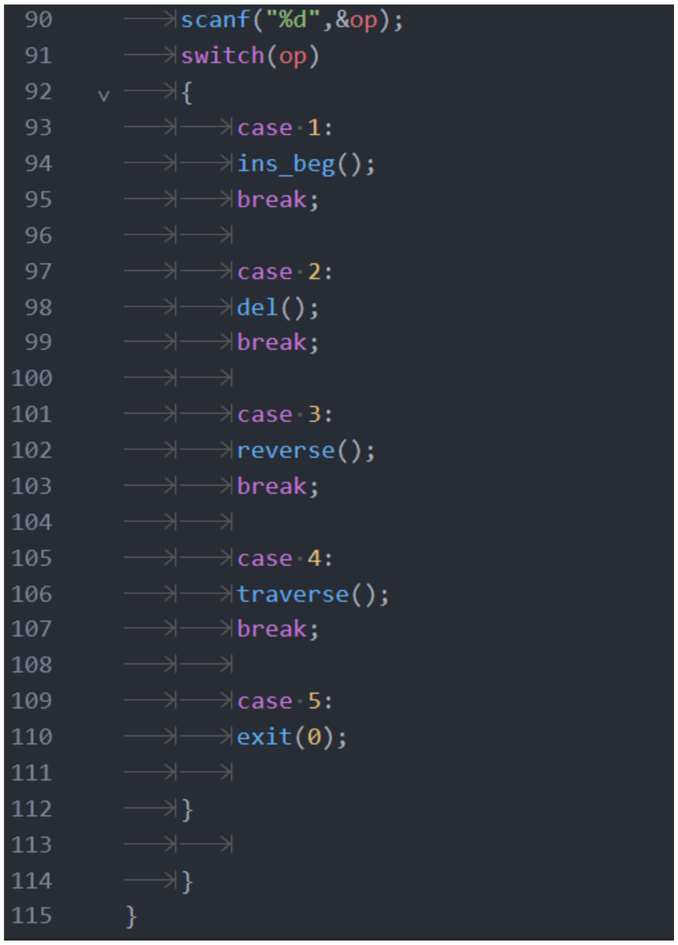
```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node *next;
};
struct node *root=NULL;
void ins_beg()
{
	struct node *temp,*p;
	temp = (struct node*)malloc(sizeof(struct node));
	printf("Enter an element : ");
	scanf("%d",&temp->data);
	temp->next = NULL;
	if(root == NULL)
	{
		root = temp;
	}
	else
	{
		
		temp->next = root;
		root = temp;
	}
}
void del()
{
	struct node *p;
```



```c
	p = root;
	if(p == NULL)
	{
		printf("Single Linked List is empty so deletion is not possible\n");
		
	}
	else
	{
		
		root = root->next;
		p->next =NULL;
		printf("The deleted element from SLL : %d\n",p->data);
		free(p);
	}
}
void traverse()
{
	
	struct node *p;
	p =root;
	if(p == NULL)
	{
		printf("Single Linked List is empty\n");
	}
	else
	{
		printf("The elements in SLL are : ");
		while(p!=NULL)
		{
			printf("%d ",p->data);
```

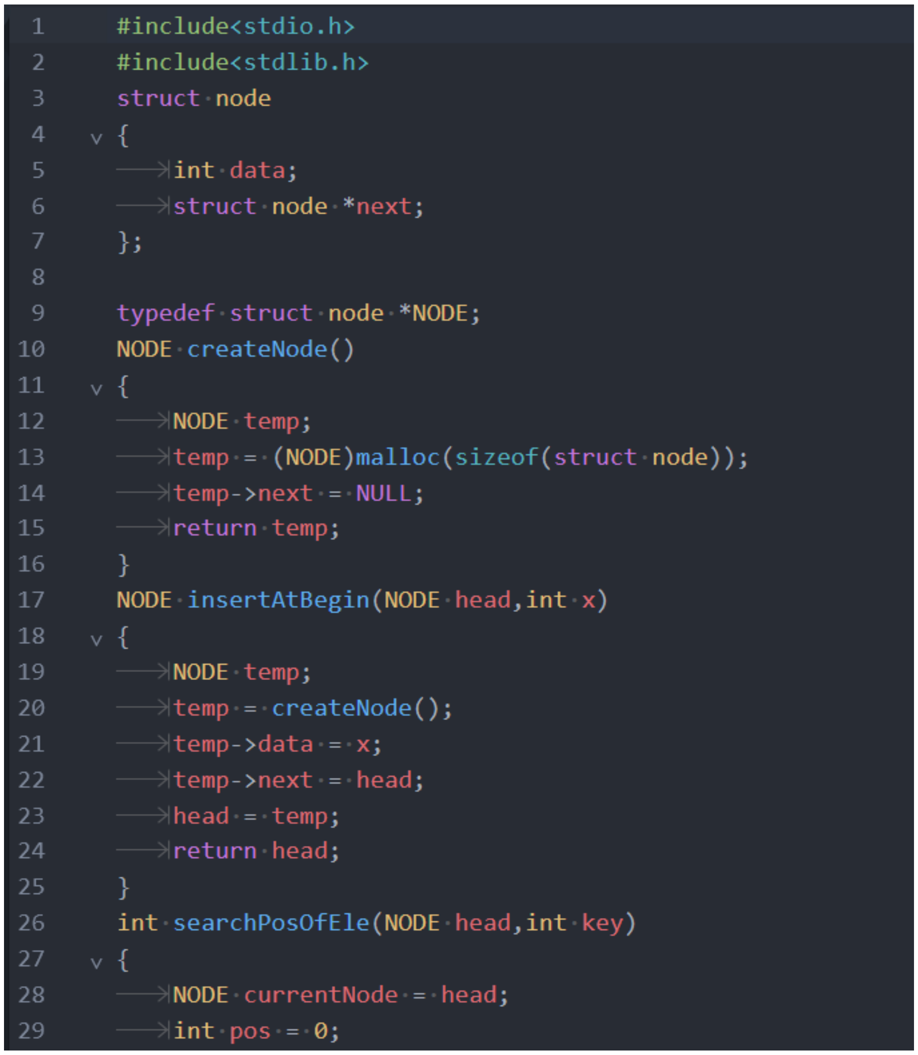


```c
            p = p->next;
        }
        printf("NULL\n");
    
        
    }
}
void reverse()
{
    struct node *ptr1,*ptr2,*ptr3;
    ptr1 = root;
    ptr2 = NULL;
    ptr3 = NULL;
    while(ptr1!=NULL)
    {
        ptr3 = ptr1->next;
        ptr1->next = ptr2;
        ptr2 = ptr1;
        ptr1 = ptr3;
    }
    root= ptr2;
    printf("Single Linked List is reversed\n");
}
int main()
{
    int op;
    while(1)
    {
        printf("1.Insert At Begin 2.Delete at Begin 3.Reverse the list 4.Traverse the List 5.Exit\n");
    printf("Enter your option : ");
```

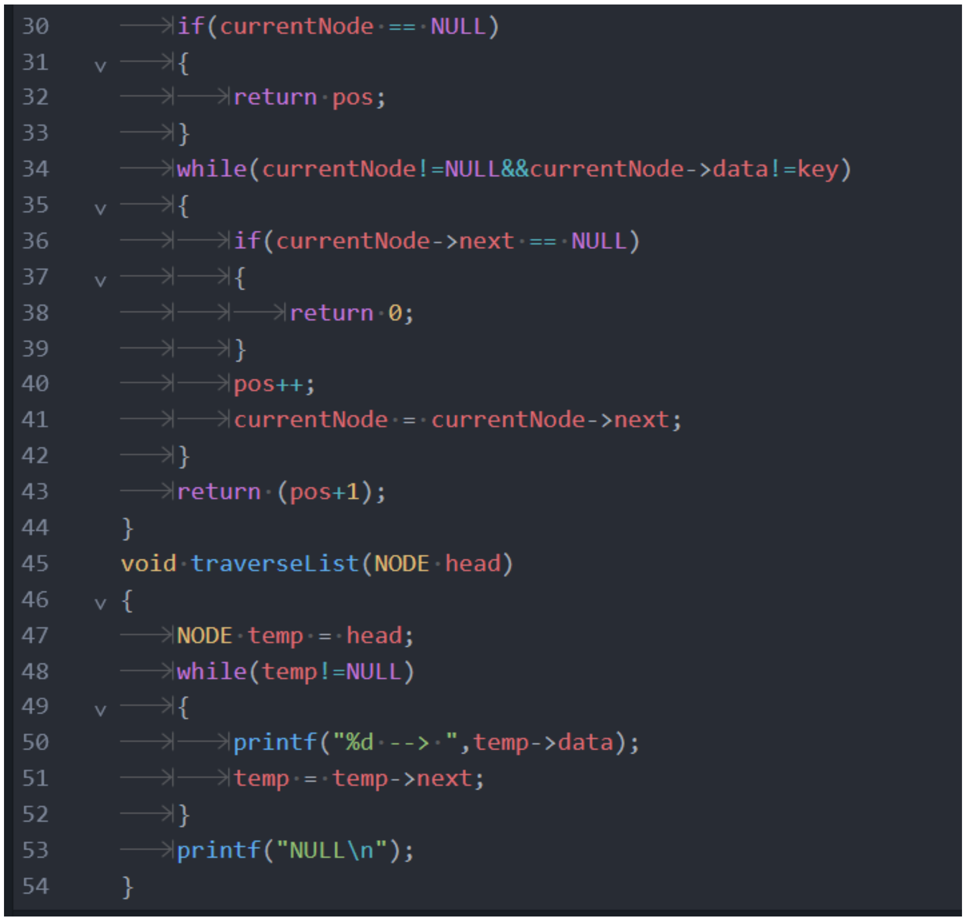


```c
	scanf("%d",&op);
	switch(op)
	{
		case 1:
		ins_beg();
		break;
		
		case 2:
		del();
		break;
		
		case 3:
		reverse();
		break;
		
		case 4:
		traverse();
		break;
		
		case 5:
		exit(0);
		
	}
		
	}
}
```

7.1.3

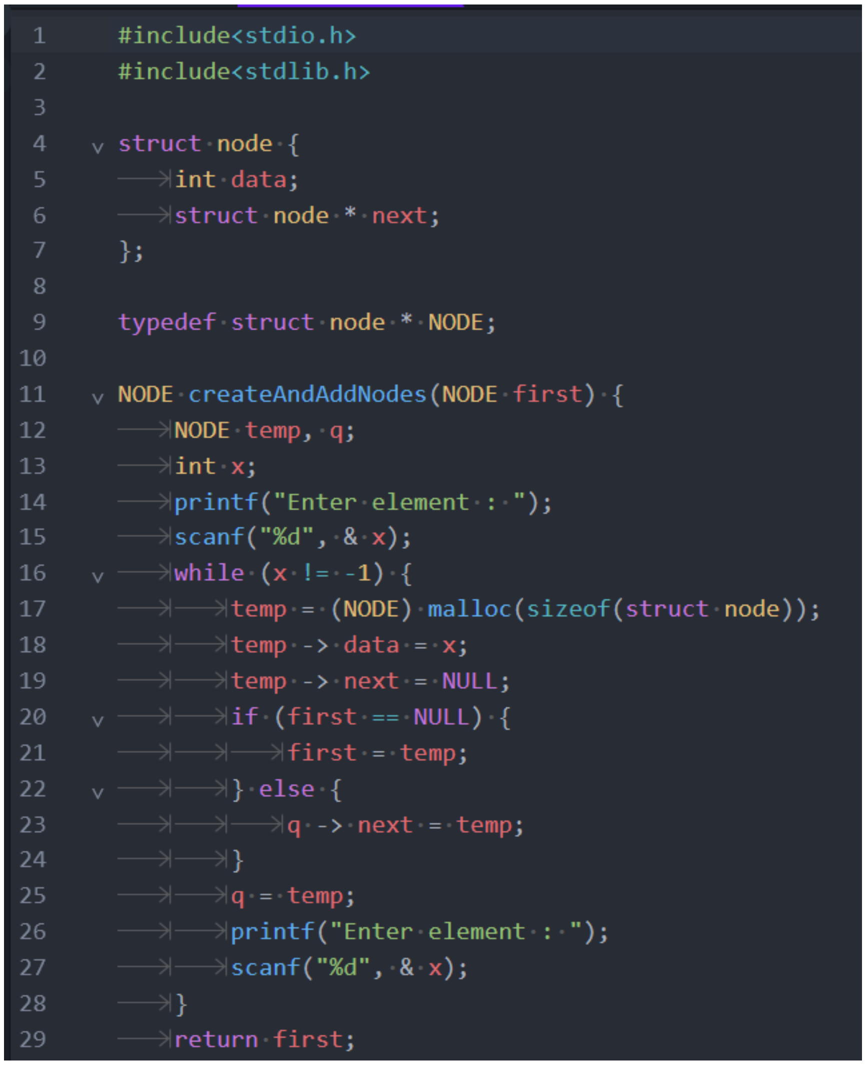


```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
	int data;
	struct node *next;
};

typedef struct node *NODE;
NODE createNode()
{
	NODE temp;
	temp = (NODE)malloc(sizeof(struct node));
	temp->next = NULL;
	return temp;
}
NODE insertAtBegin(NODE head,int x)
{
	NODE temp;
	temp = createNode();
	temp->data = x;
	temp->next = head;
	head = temp;
	return head;
}
int searchPosOfEle(NODE head,int key)
{
	NODE currentNode = head;
	int pos = 0;
```

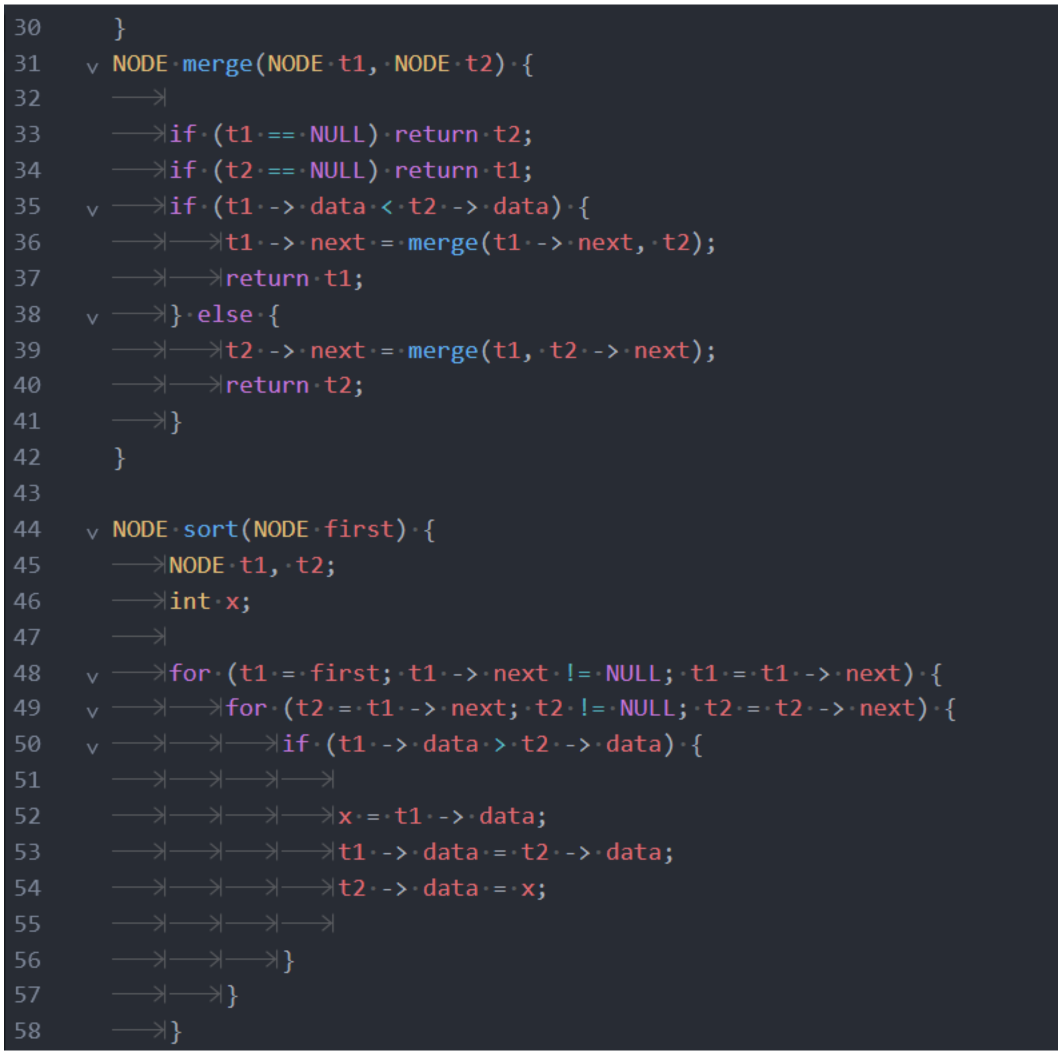
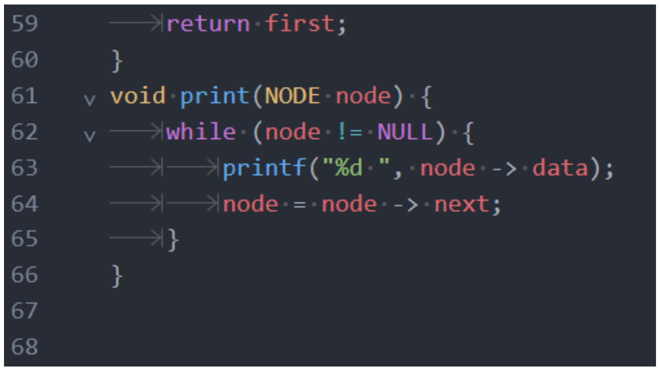


```
	if(currentNode == NULL)
	{
		return pos;
	}
	while(currentNode!=NULL&&currentNode->data!=key)
	{
		if(currentNode->next == NULL)
		{
			return 0;
		}
		pos++;
		currentNode = currentNode->next;
	}
	return (pos+1);
}
void traverseList(NODE head)
{
	NODE temp = head;
	while(temp!=NULL)
	{
		printf("%d --> ",temp->data);
		temp = temp->next;
	}
	printf("NULL\n");
}
```

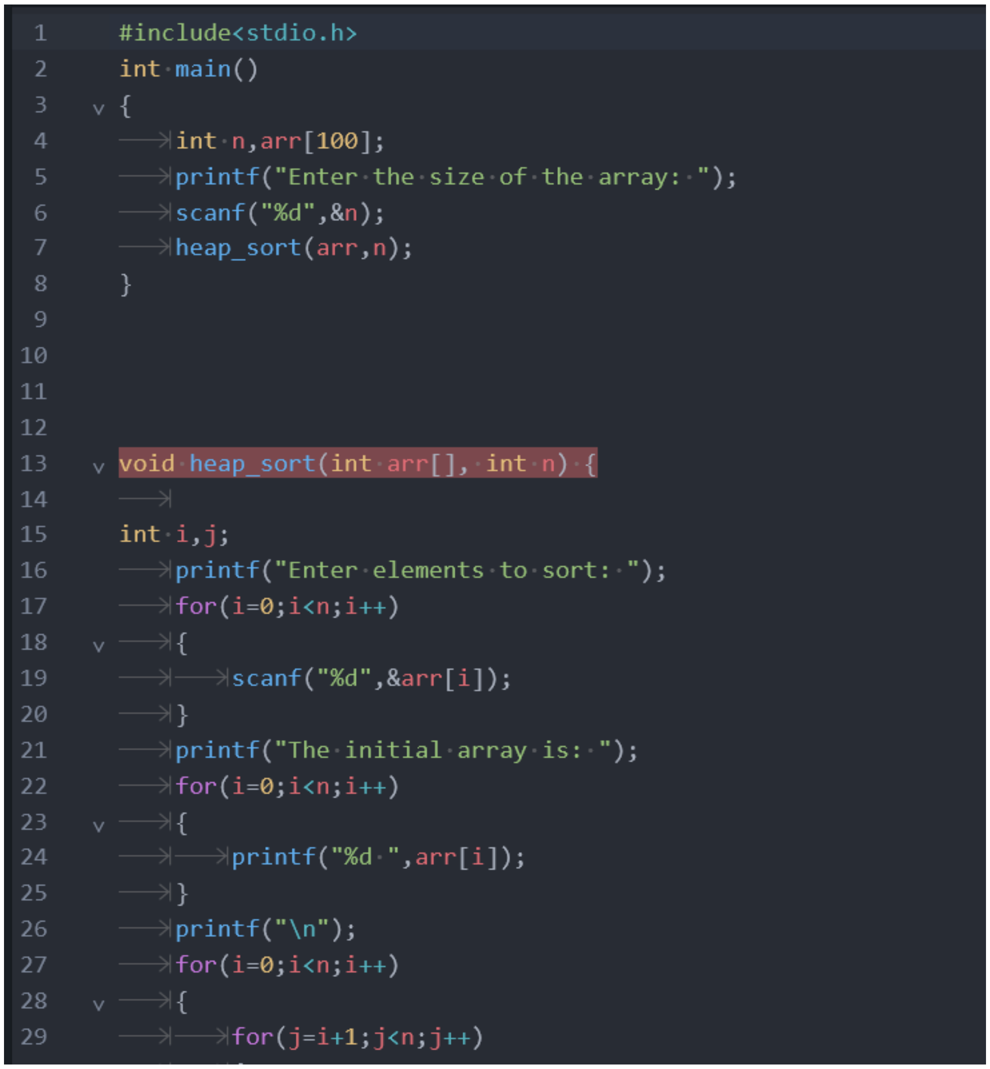
7.1.4



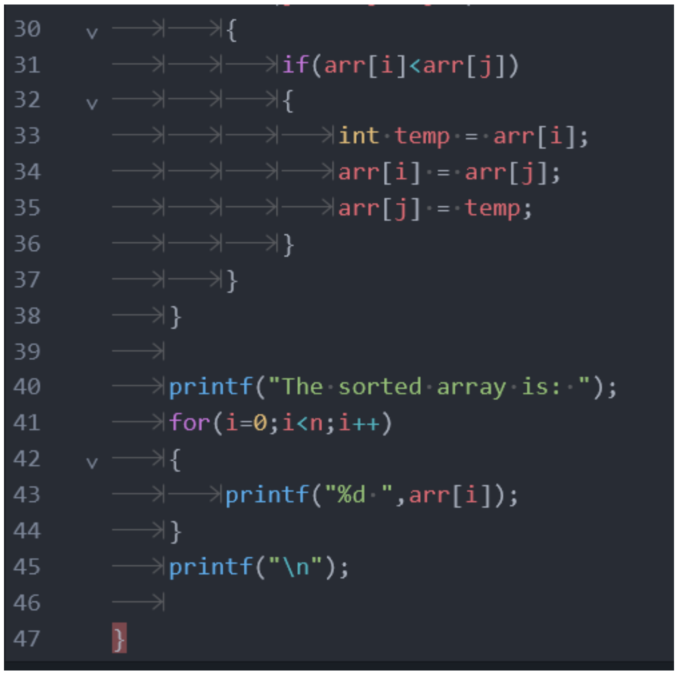
```c
#include<stdio.h>
#include<stdlib.h>

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

typedef struct node * NODE;

NODE createAndAddNodes(NODE first) {
	NODE temp, q;
	int x;
	printf("Enter element : ");
	scanf("%d", & x);
	while (x != -1) {
		temp = (NODE) malloc(sizeof(struct node));
		temp -> data = x;
		temp -> next = NULL;
		if (first == NULL) {
			first = temp;
		} else {
			q -> next = temp;
		}
		q = temp;
		printf("Enter element : ");
		scanf("%d", & x);
	}
	return first;
```


```c
}
NODE merge(NODE t1, NODE t2) {
	
	if (t1 == NULL) return t2;
	if (t2 == NULL) return t1;
	if (t1 -> data < t2 -> data) {
		t1 -> next = merge(t1 -> next, t2);
		return t1;
	} else {
		t2 -> next = merge(t1, t2 -> next);
		return t2;
	}
}

NODE sort(NODE first) {
	NODE t1, t2;
	int x;
	
	for (t1 = first; t1 -> next != NULL; t1 = t1 -> next) {
		for (t2 = t1 -> next; t2 != NULL; t2 = t2 -> next) {
			if (t1 -> data > t2 -> data) {
				
				x = t1 -> data;
				t1 -> data = t2 -> data;
				t2 -> data = x;
				
			}
		}
	}
```

```c
	return first;
}
void print(NODE node) {
	while (node != NULL) {
		printf("%d ", node -> data);
		node = node -> next;
	}
}


```

7.2.1

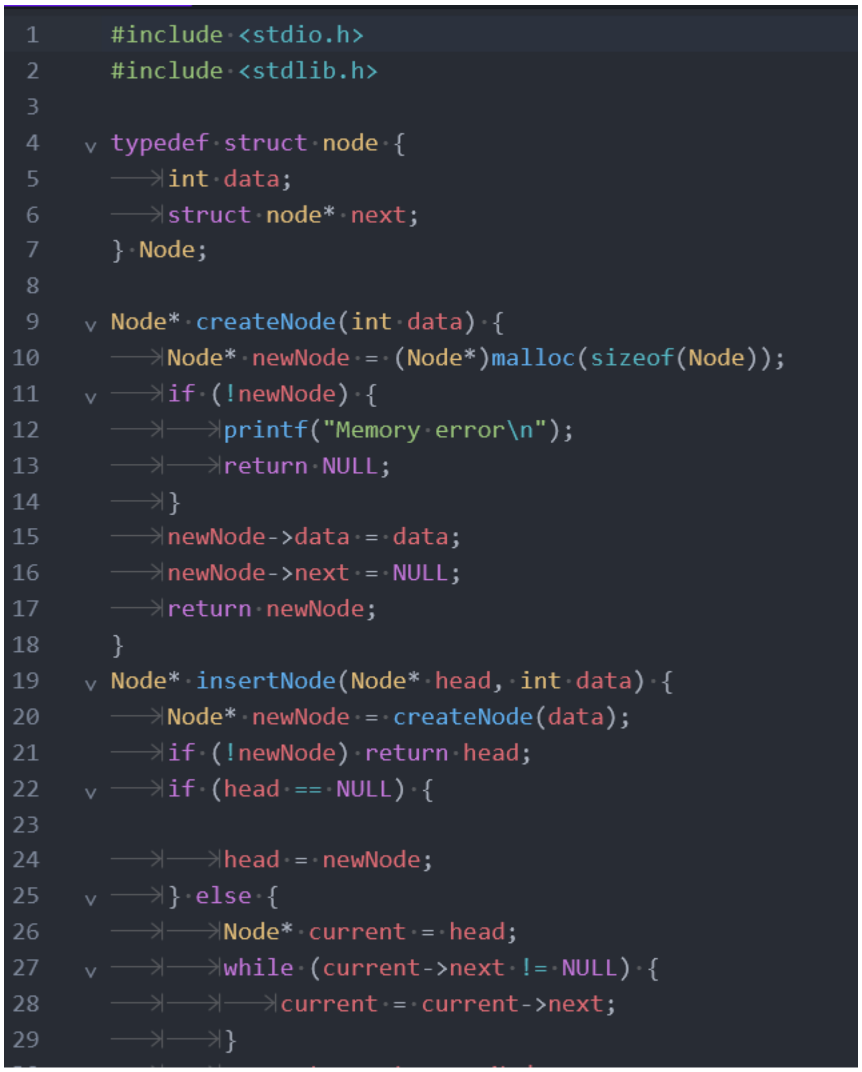


```c
#include<stdio.h>
int main()
{
	int n,arr[100];
	printf("Enter the size of the array: ");
	scanf("%d",&n);
	heap_sort(arr,n);
}




void heap_sort(int arr[], int n) {
	
int i,j;
	printf("Enter elements to sort: ");
	for(i=0;i<n;i++)
	{
		scanf("%d",&arr[i]);
	}
	printf("The initial array is: ");
	for(i=0;i<n;i++)
	{
		printf("%d ",arr[i]);
	}
	printf("\n");
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<n;j++)
```

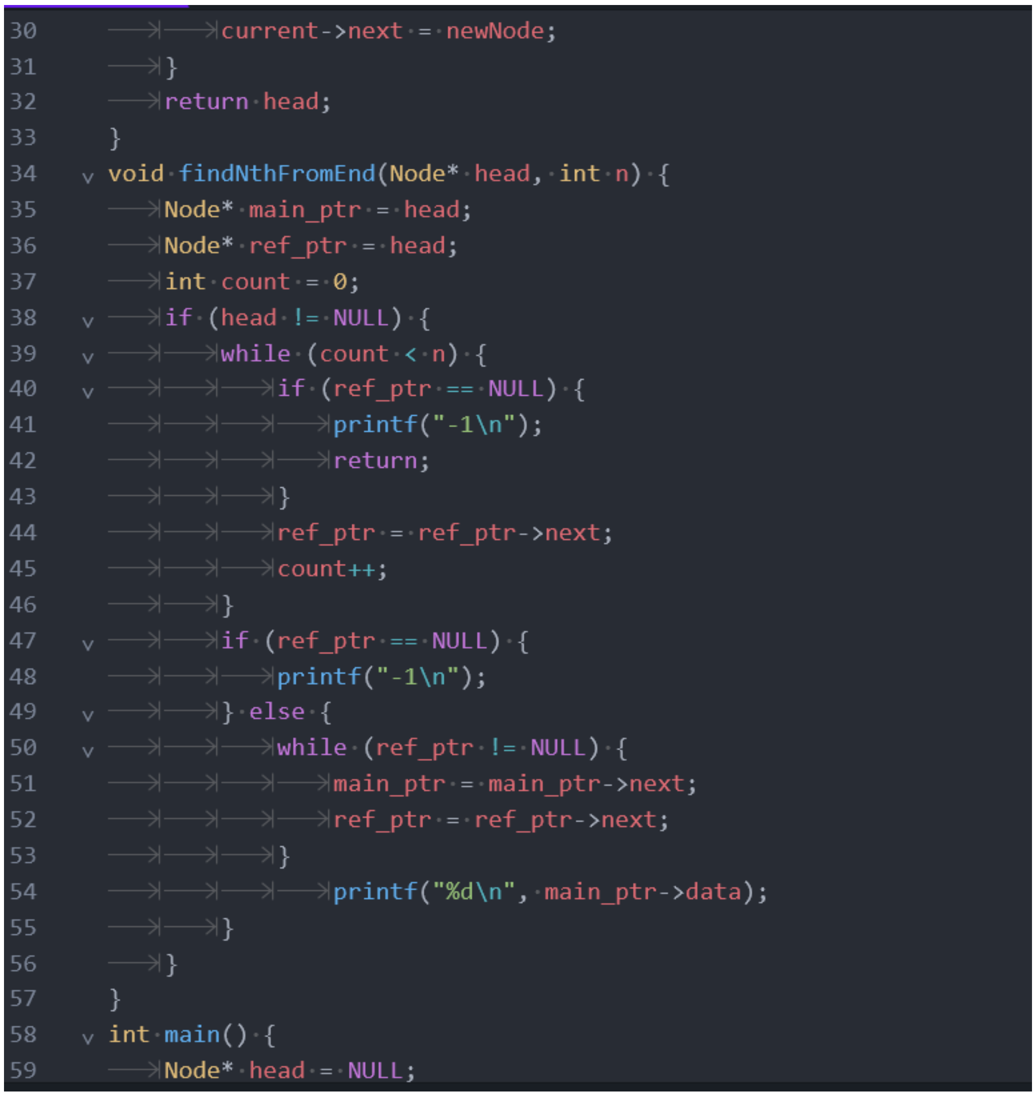


```
        {
            if(arr[i]<arr[j])
            {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
    
    printf("The sorted array is: ");
    for(i=0;i<n;i++)
    {
        printf("%d ",arr[i]);
    }
    printf("\n");
    
}
```

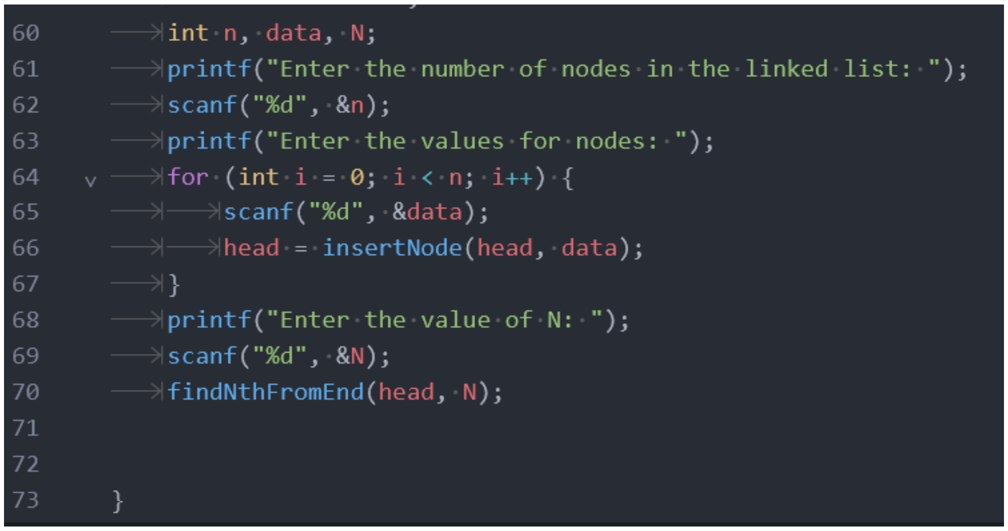
7.2.2



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

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

Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) {
        printf("Memory error\n");
        return NULL;
    }
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}
Node* insertNode(Node* head, int data) {
    Node* newNode = createNode(data);
    if (!newNode) return head;
    if (head == NULL) {

        head = newNode;
    } else {
        Node* current = head;
        while (current->next != NULL) {
            current = current->next;
        }
```



```c
        current->next = newNode;
    }
    return head;
}
void findNthFromEnd(Node* head, int n) {
    Node* main_ptr = head;
    Node* ref_ptr = head;
    int count = 0;
    if (head != NULL) {
        while (count < n) {
            if (ref_ptr == NULL) {
                printf("-1\n");
                return;
            }
            ref_ptr = ref_ptr->next;
            count++;
        }
        if (ref_ptr == NULL) {
            printf("-1\n");
        } else {
            while (ref_ptr != NULL) {
                main_ptr = main_ptr->next;
                ref_ptr = ref_ptr->next;
            }
                printf("%d\n", main_ptr->data);
        }
    }
}
int main() {
    Node* head = NULL;
```



```c
    int n, data, N;
    printf("Enter the number of nodes in the linked list: ");
    scanf("%d", &n);
    printf("Enter the values for nodes: ");
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        head = insertNode(head, data);
    }
    printf("Enter the value of N: ");
    scanf("%d", &N);
    findNthFromEnd(head, N);


}
```