DS Lab Program 9 -

```
#include <stdio.h>
#include <stdlib.h>.
struct Node {
  int info ;
   struct Node *rlink, * llink;
};
typedef struct Node node;
node getnode () {
   node x;
   x= (node) malloc (sizeof (node));
   if (x==NULL) {
       printf ("Memory full \n");
       exit (0);
   }
   return x;
}
void freenode (node x) {
         free(x);
 }.
node insert_front (int item, node head) {
   node temp, cur;
   temp= getnode ();
   temp -> info = item;
   ~~temp -> rlink = null~~;
   cur = head -> rlink;
   ~~head = head -> rlink~~
   head-> rlink = head;
   temp -> rlink = cur;
   cur -> llink= Temp;
   return head;
}
```

Date / /

```
node insert_rear (int item, node head) {
  node temp = getnode (), cur;
  temp -> ~~[illegible]~~ info = item;
  cur = head -> llink;
  head -> llink = head;
  temp -> rlink = head;
  temp -> llink = cur;
  cur -> rlink = temp;
  return head;
}
node delete_front (node head) {
  node cur, next;
  if (head -> rlink == head) {
    printf ("Empty \n");
    return head;
  }
  cur = head -> rlink;
  next = cur -> rlink;
  head -> rlink = next;
  next -> llink = head;
  printf ("Item deleted = %d\n", cur -> info);
  freenode (cur);
  return head;
}
node delete_rear (node head) {
  node cur, prev;
  if (head -> rlink == head) {
    printf ("Empty \n");
    return head;
  }
  cur = head -> llink;
  prev = cur -> llink;
  head -> llink = prev;
```

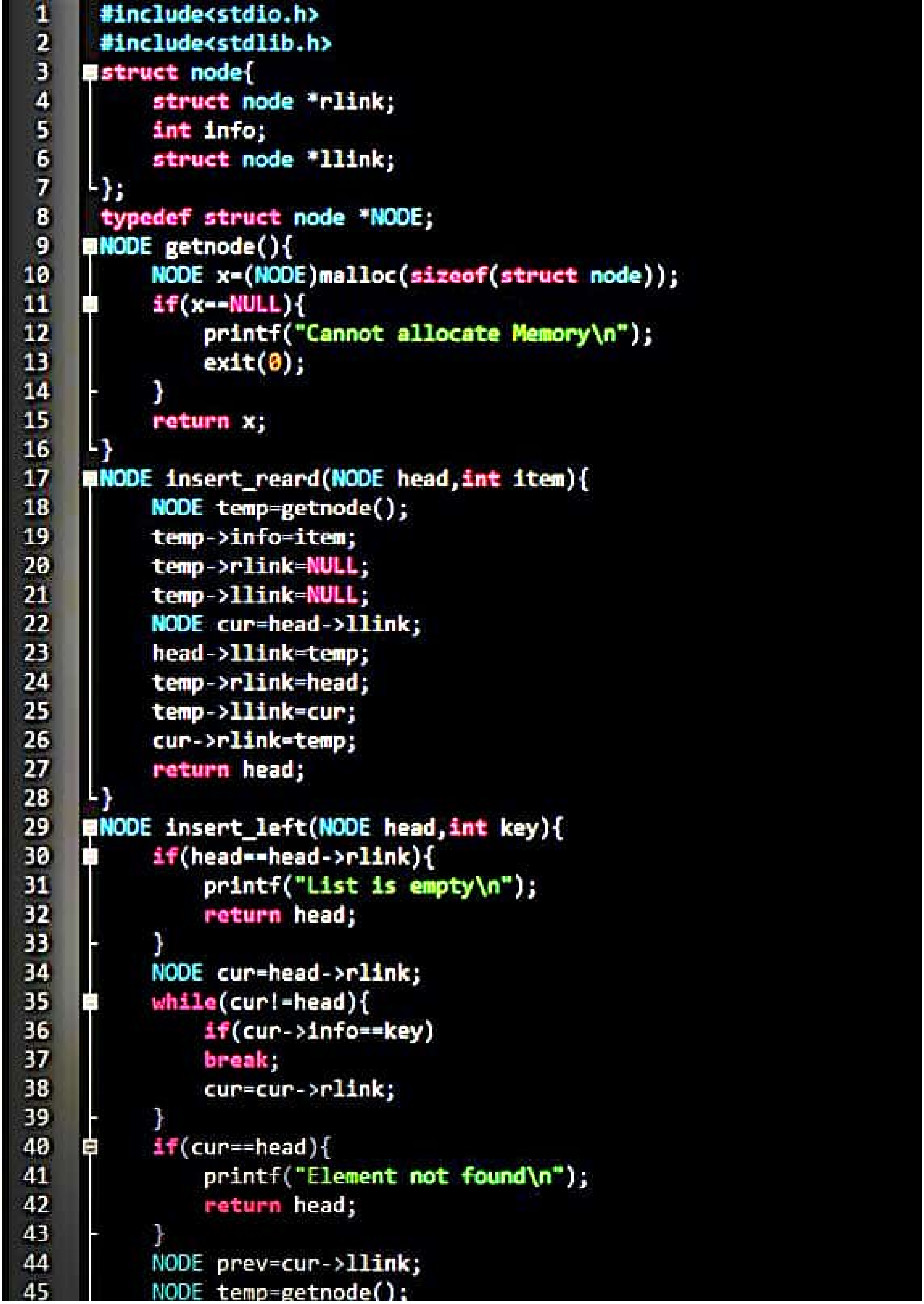
Scanned with CamScanner

```
    prev -> rlink = head;
    printf ("Item deleted = %d\n", cur -> info);
    freenode (cur);
    return head;
}
node insert leftpos (int item, node head) {
    node temp, cur, prev;
    if (head -> rlink == head) {
        printf (" Empty \n");
        return head;
    }
    cur = head -> rlink;
    while (cur != head) {
        if (item == cur -> info)
            break;
        cur = cur -> rlink;
    }
    if (cur == head) {
        printf ("Key not found \n");
        return head.
    }
    prev = cur -> llink;
    printf(" Enter towards left of %d", item);
    Temp = getnode ();
    scanf (" %d", temp -> info);
    prev -> rlink = temp;
    temp -> llink = prev;
    cur -> llink = temp;
    temp -> rlink = cur;
    return head;
}
node delete_value (int item, node head) {
```

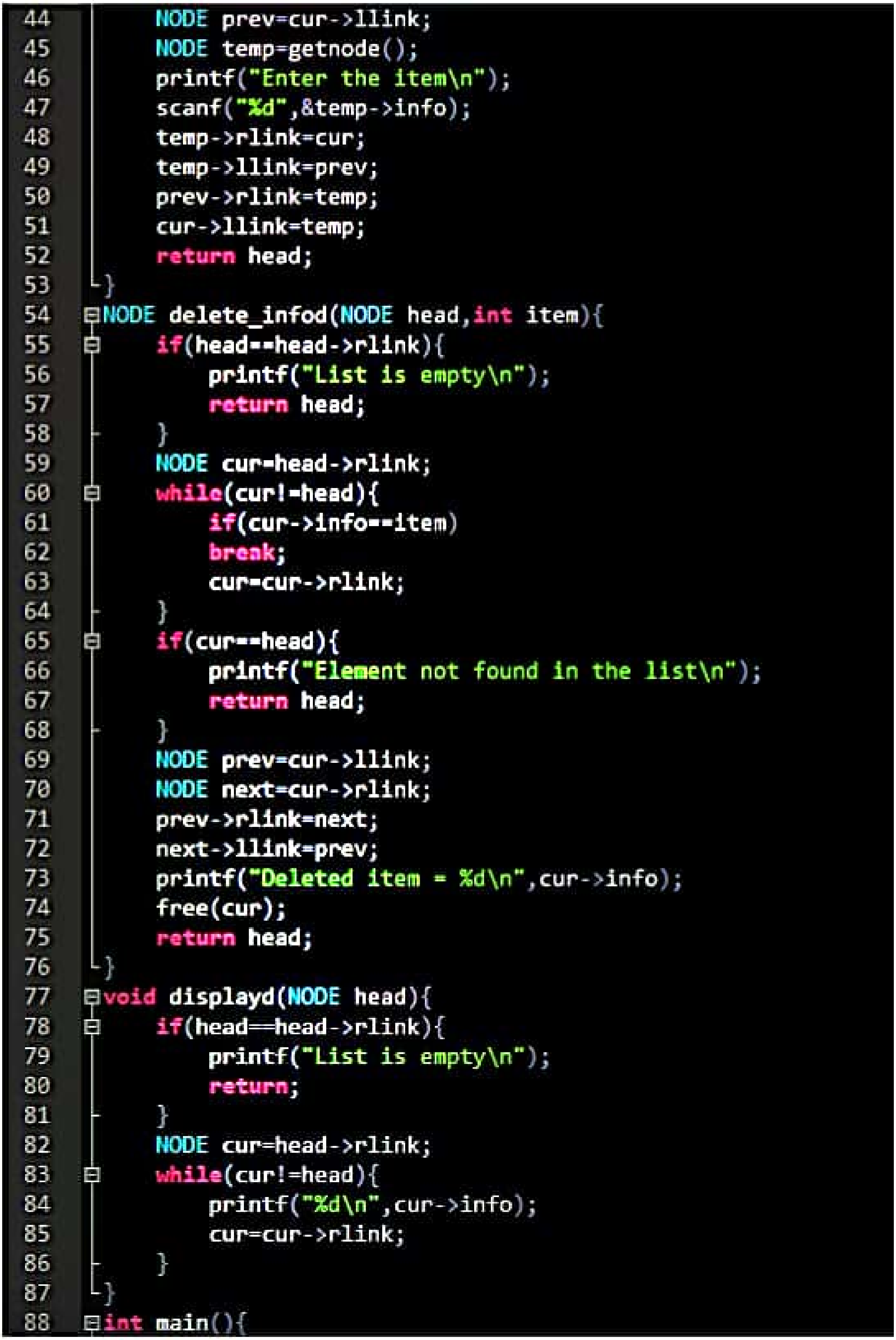
Scanned with CamScanner

```
node prev, cur, next;
int count;
if (head -> rlink == head) {
    printf ("List is empty \n");
    return head;
}
count = 0;
cur = head -> rlink;
while (cur != head) {
    if (item != cur -> info)
        cur = cur -> rlink;
    else {
        count ++;
        prev = cur -> llink ;
        next = cur -> rlink;
        prev -> rlink = next ;
        next -> llink = prev;
        freenode (cur);
        }
  }
  if (count == 0)
      printf ("Key not found \n");
  else
      printf (" Key found at % position and is
                    deleted \n", count);
  return head;
}
void display (node head) {
  node temp;
  if (head -> rlink == head) {
      printf ("Empty \n");
      return;
  }
```

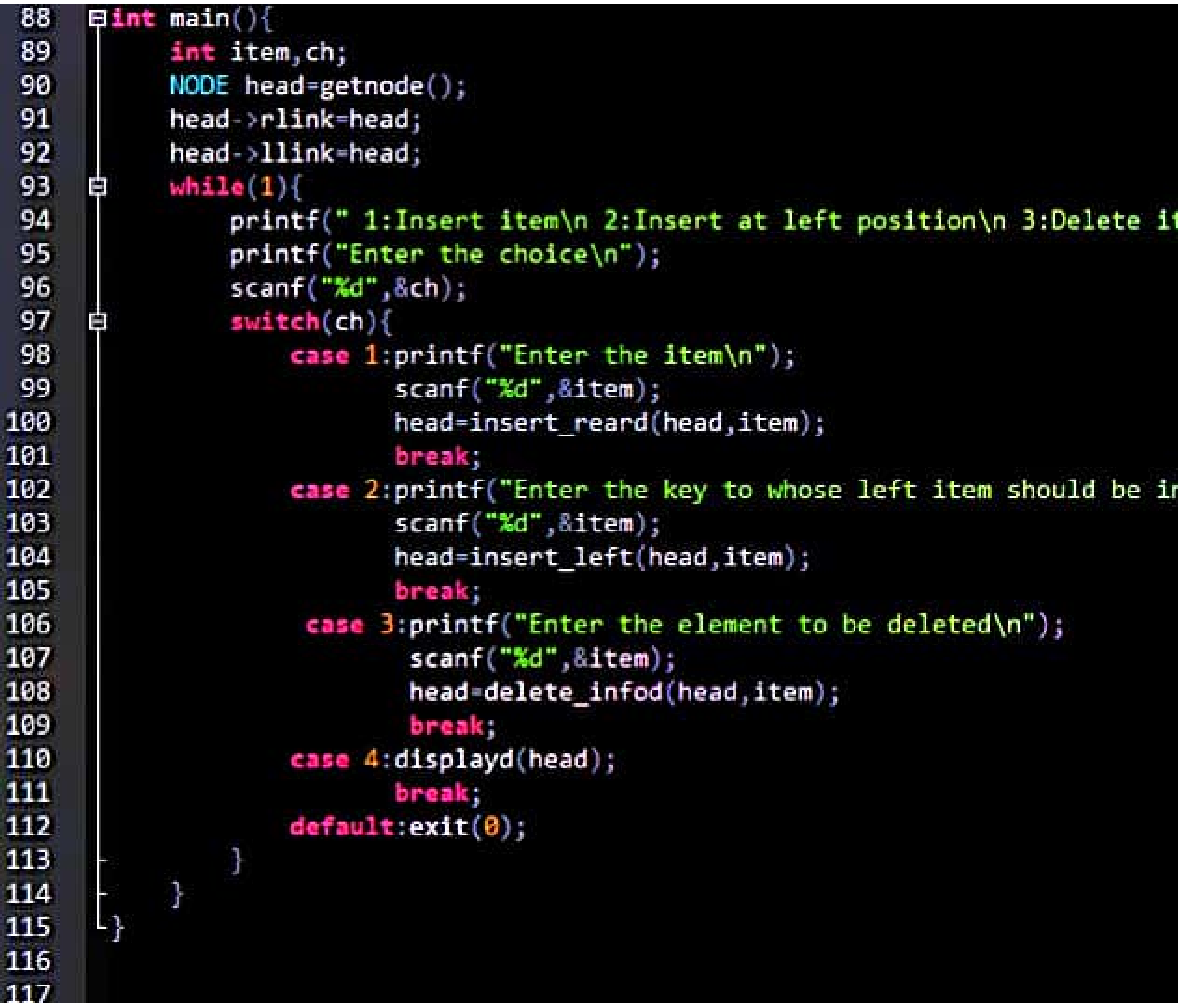
```
printf ("Contents of Queue \n");
temp = head -> rlink;
while (temp != head) {
    printf ("%d\n", temp -> info);
    temp = temp -> rlink;
  }
}
void main () {
node head, last;
int item, choice;
head = getnode();
head -> rlink = head;
head -> llink = head;
for (;;) {
  printf ("\n1. Insert front \n2. Insert rear \n3
  Delete front \n4. Delete Rear \n 5. Insert left
  position \n 6. Delete specified value \n7. Display\n");
  scanf ("%d", &choice);
  switch (choice) {
  case1: printf ("Enter item \n");
         scanf ("%d", & item);
         last = insert_front (item, head);
         break;
  case2: printf ("Enter item \n");
         scanf ("%d", & item);
         last = insert_rear (item, head);
         break;
  case3: last = delete_front (head);
         break;
  case4: last = delete_rear (head);
         break;
  case 5: printf (" Enter key item \n");
          scanf ("%d", & item);
head = insert_leftpos (item, head);
break;
case 6: printf ("Enter key item\n");
        scanf ("%d", & item);
        head = delete_value (item, head);
        break;
case 7: display (head);
        break;
default: exit (0);
} } }.
```



```c
#include<stdio.h>
#include<stdlib.h>
struct node{
	struct node *rlink;
	int info;
	struct node *llink;
};
typedef struct node *NODE;
NODE getnode(){
	NODE x=(NODE)malloc(sizeof(struct node));
	if(x==NULL){
		printf("Cannot allocate Memory\n");
		exit(0);
	}
	return x;
}
NODE insert_reard(NODE head,int item){
	NODE temp=getnode();
	temp->info=item;
	temp->rlink=NULL;
	temp->llink=NULL;
	NODE cur=head->llink;
	head->llink=temp;
	temp->rlink=head;
	temp->llink=cur;
	cur->rlink=temp;
	return head;
}
NODE insert_left(NODE head,int key){
	if(head==head->rlink){
		printf("List is empty\n");
		return head;
	}
	NODE cur=head->rlink;
	while(cur!=head){
		if(cur->info==key)
		break;
		cur=cur->rlink;
	}
	if(cur==head){
		printf("Element not found\n");
		return head;
	}
	NODE prev=cur->llink;
	NODE temp=getnode();
```

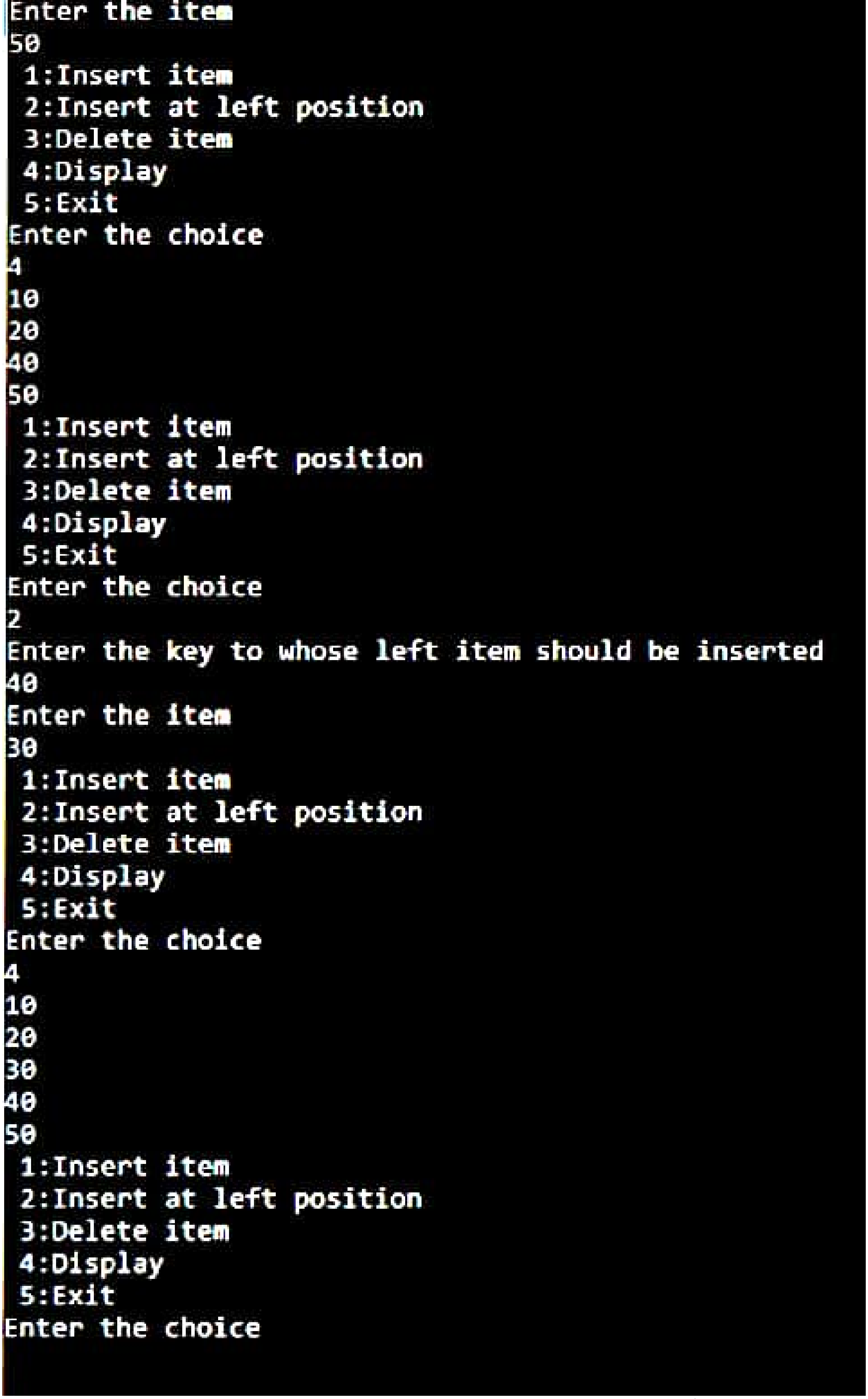


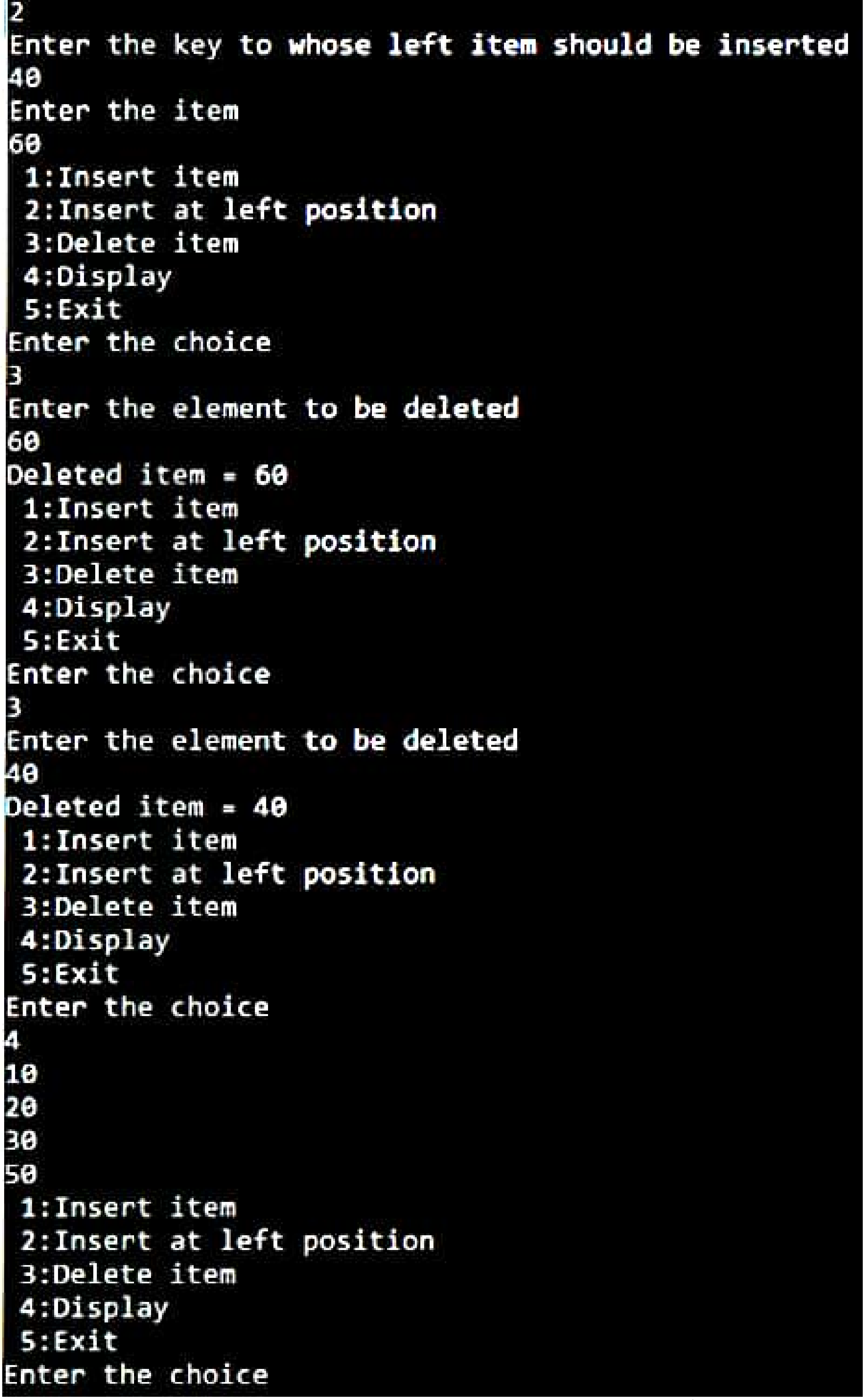
```c
    NODE prev=cur->llink;
    NODE temp=getnode();
    printf("Enter the item\n");
    scanf("%d",&temp->info);
    temp->rlink=cur;
    temp->llink=prev;
    prev->rlink=temp;
    cur->llink=temp;
    return head;
}
NODE delete_infod(NODE head,int item){
    if(head==head->rlink){
        printf("List is empty\n");
        return head;
    }
    NODE cur=head->rlink;
    while(cur!=head){
        if(cur->info==item)
        break;
        cur=cur->rlink;
    }
    if(cur==head){
        printf("Element not found in the list\n");
        return head;
    }
    NODE prev=cur->llink;
    NODE next=cur->rlink;
    prev->rlink=next;
    next->llink=prev;
    printf("Deleted item = %d\n",cur->info);
    free(cur);
    return head;
}
void displayd(NODE head){
    if(head==head->rlink){
        printf("List is empty\n");
        return;
    }
    NODE cur=head->rlink;
    while(cur!=head){
        printf("%d\n",cur->info);
        cur=cur->rlink;
    }
}
int main(){
```



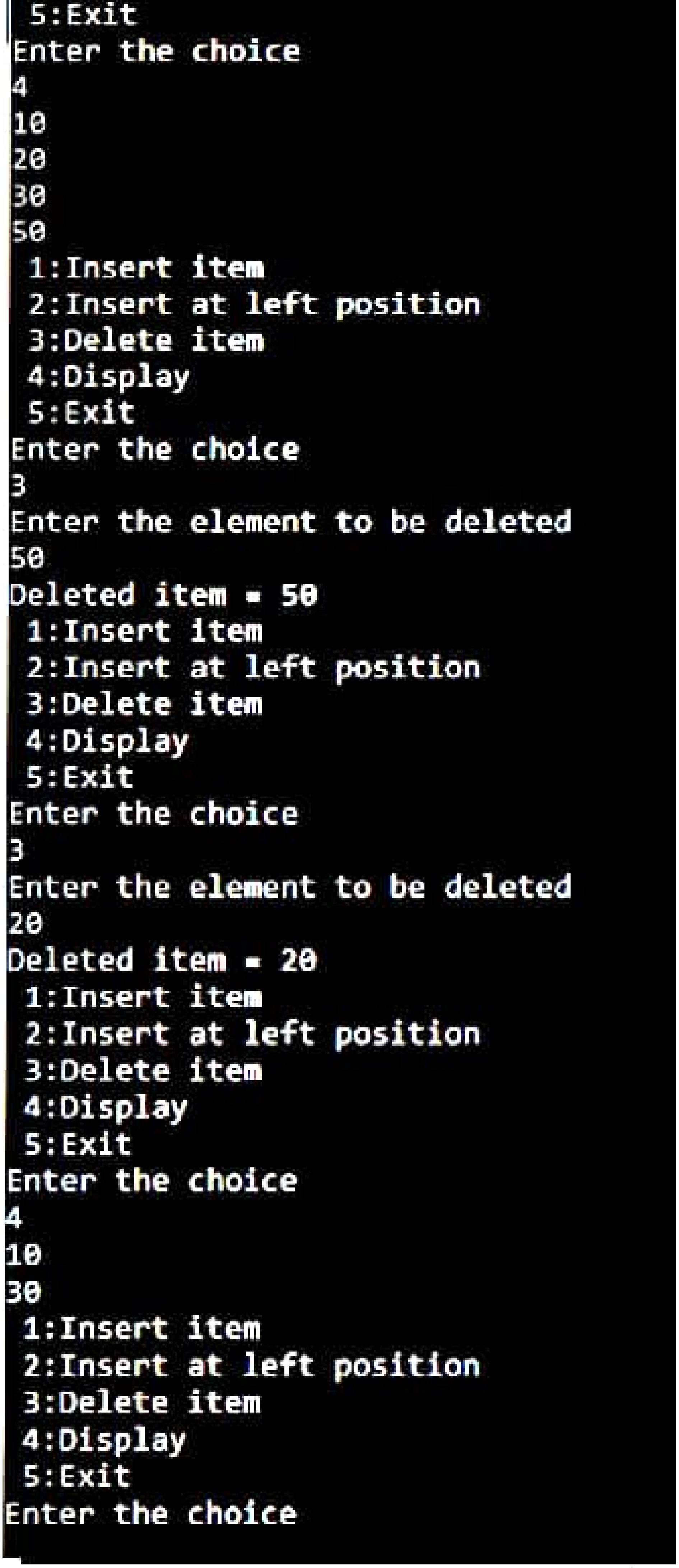
```
int main(){
    int item,ch;
    NODE head=getnode();
    head->rlink=head;
    head->llink=head;
    while(1){
        printf(" 1:Insert item\n 2:Insert at left position\n 3:Delete it
        printf("Enter the choice\n");
        scanf("%d",&ch);
        switch(ch){
            case 1:printf("Enter the item\n");
                   scanf("%d",&item);
                   head=insert_reard(head,item);
                   break;
            case 2:printf("Enter the key to whose left item should be in
                   scanf("%d",&item);
                   head=insert_left(head,item);
                   break;
             case 3:printf("Enter the element to be deleted\n");
                    scanf("%d",&item);
                    head=delete_infod(head,item);
                    break;
            case 4:displayd(head);
                   break;
            default:exit(0);
        }
    }
}
```

Scanned with CamScanner


```
Enter the item
50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
4
10
20
40
50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
2
Enter the key to whose left item should be inserted
40
Enter the item
30
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
4
10
20
30
40
50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
```



```
2
Enter the key to whose left item should be inserted
40
Enter the item
60
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
3
Enter the element to be deleted
60
Deleted item = 60
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
3
Enter the element to be deleted
40
Deleted item = 40
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
4
10
20
30
50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
```



```
 5:Exit
Enter the choice
4
10
20
30
50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
3
Enter the element to be deleted
50
Deleted item = 50
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
3
Enter the element to be deleted
20
Deleted item = 20
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
4
10
30
 1:Insert item
 2:Insert at left position
 3:Delete item
 4:Display
 5:Exit
Enter the choice
```