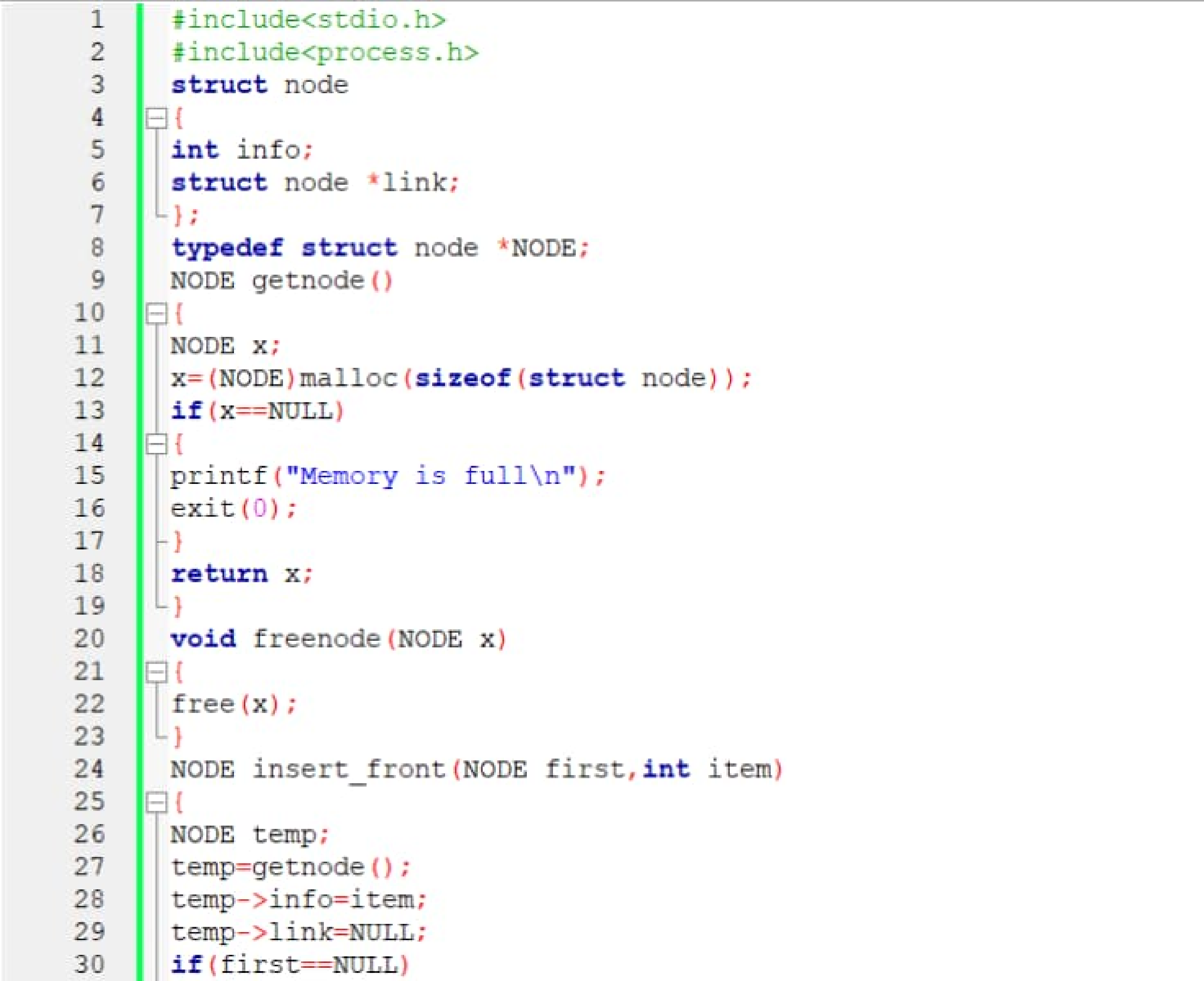
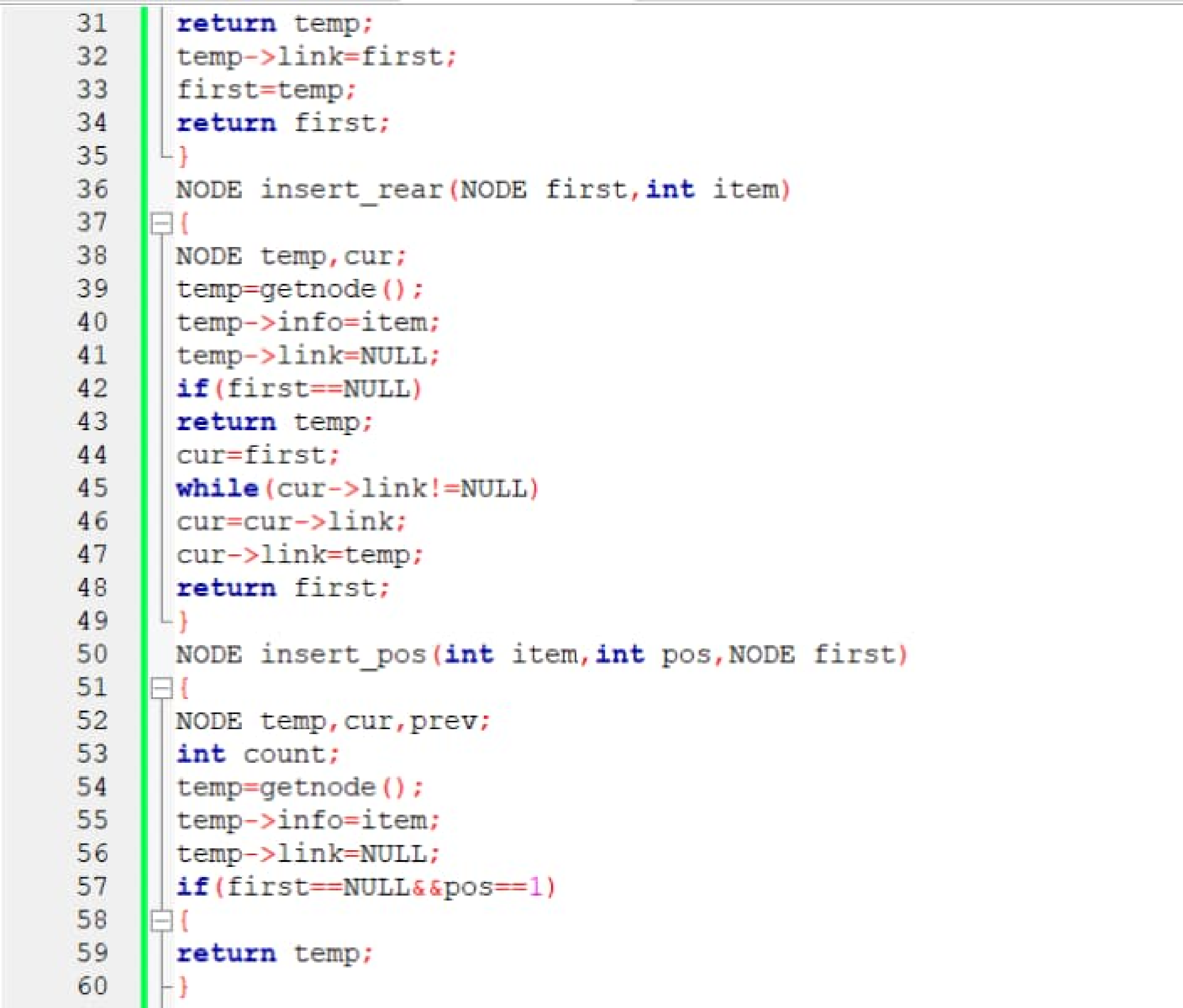
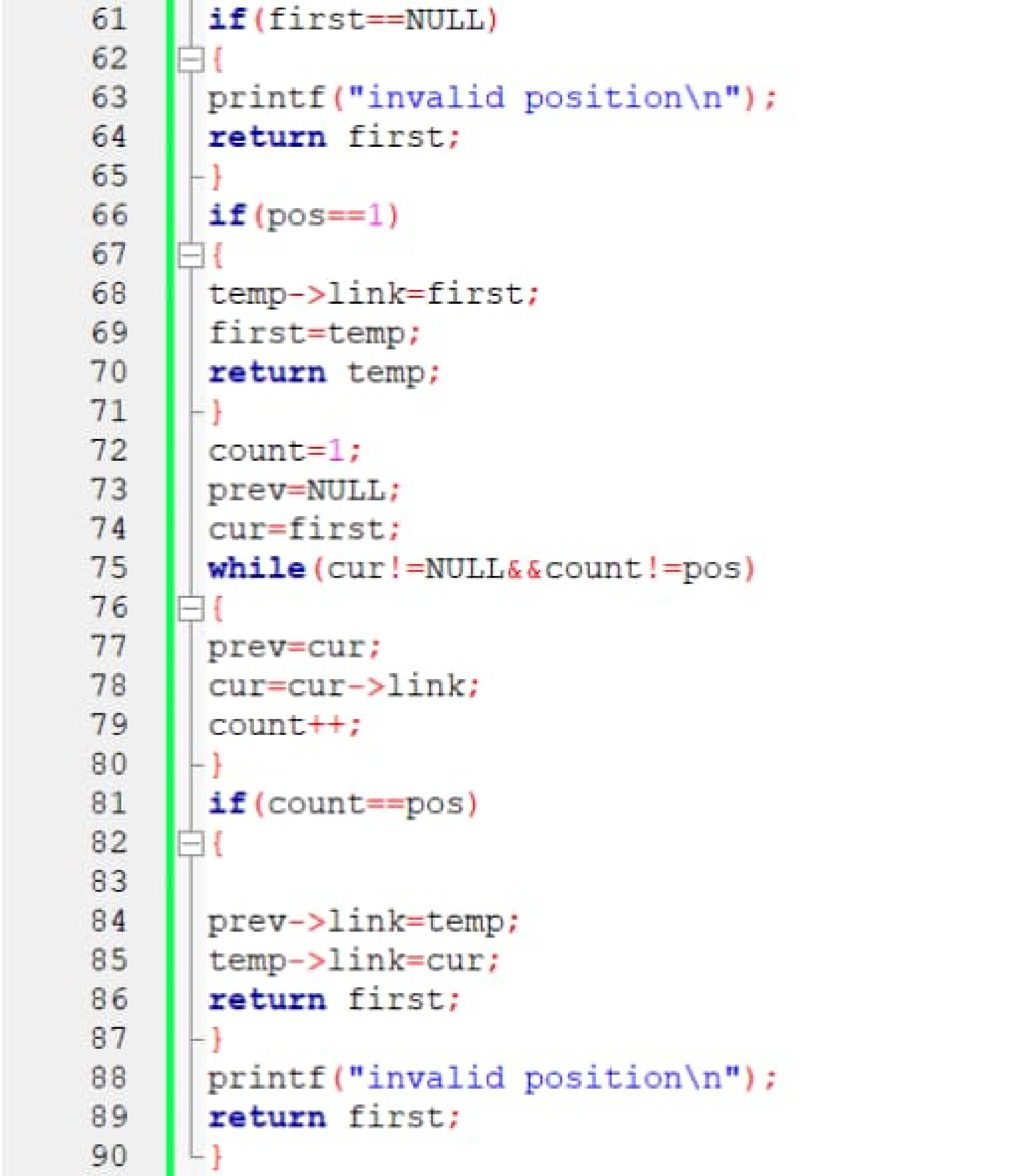


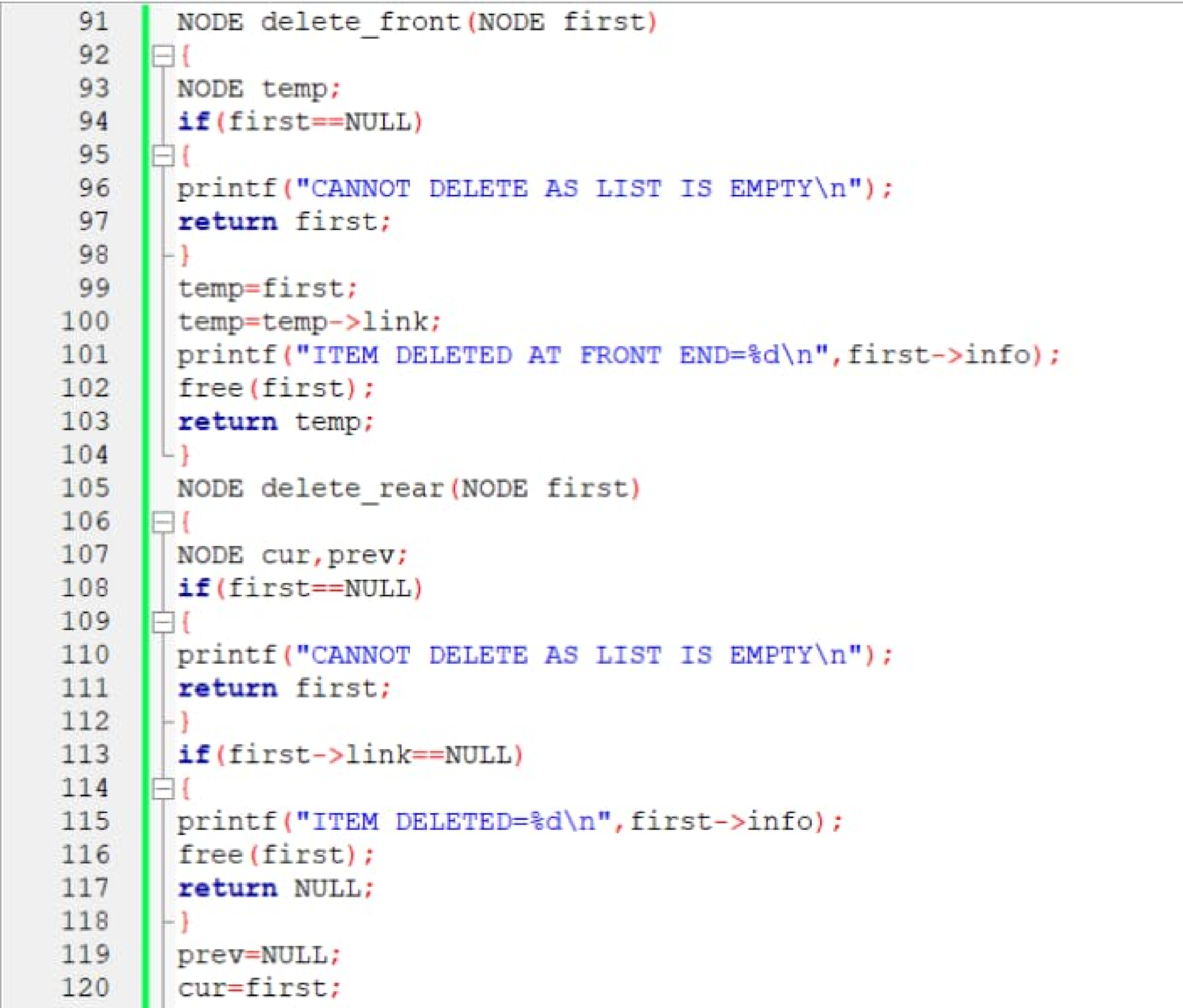
```c
#include<stdio.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("Memory is full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
```



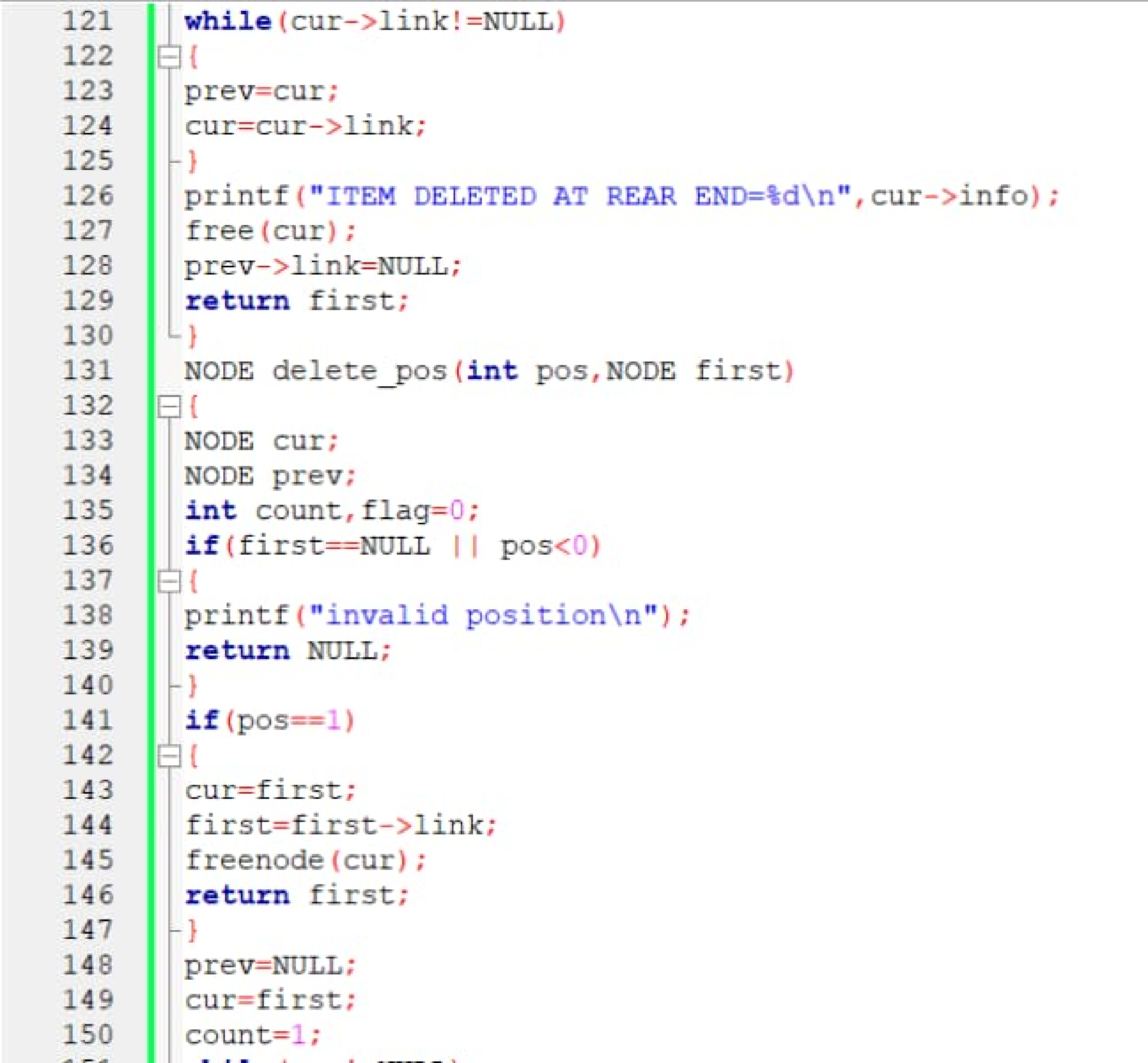
```c
return temp;
temp->link=first;
first=temp;
return first;
}
NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
return temp;
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
return first;
}
NODE insert_pos(int item,int pos,NODE first)
{
NODE temp,cur,prev;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL&&pos==1)
{
return temp;
}
```



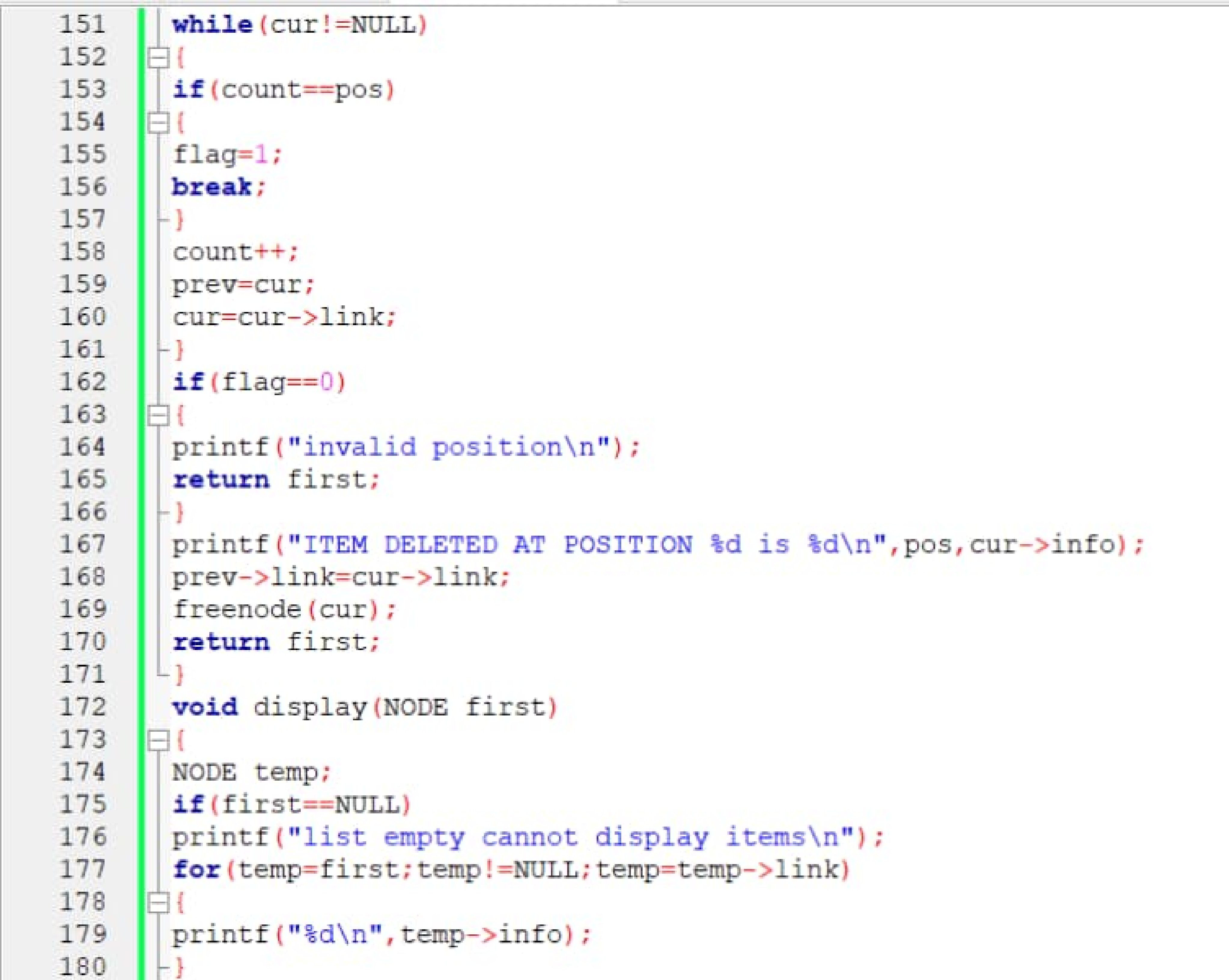
```
if(first==NULL)
{
printf("invalid position\n");
return first;
}
if(pos==1)
{
temp->link=first;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL&&count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{

prev->link=temp;
temp->link=cur;
return first;
}
printf("invalid position\n");
return first;
}
```



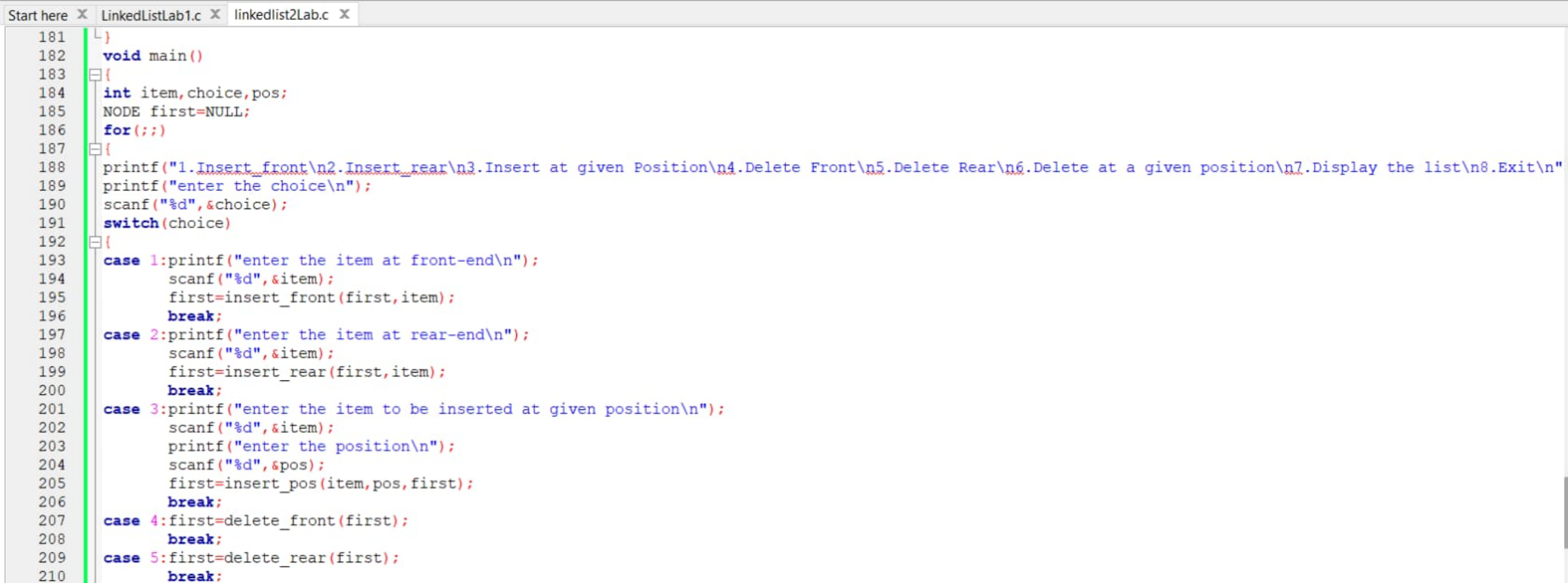
```c
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("CANNOT DELETE AS LIST IS EMPTY\n");
return first;
}
temp=first;
temp=temp->link;
printf("ITEM DELETED AT FRONT END=%d\n",first->info);
free(first);
return temp;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("CANNOT DELETE AS LIST IS EMPTY\n");
return first;
}
if(first->link==NULL)
{
printf("ITEM DELETED=%d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
```



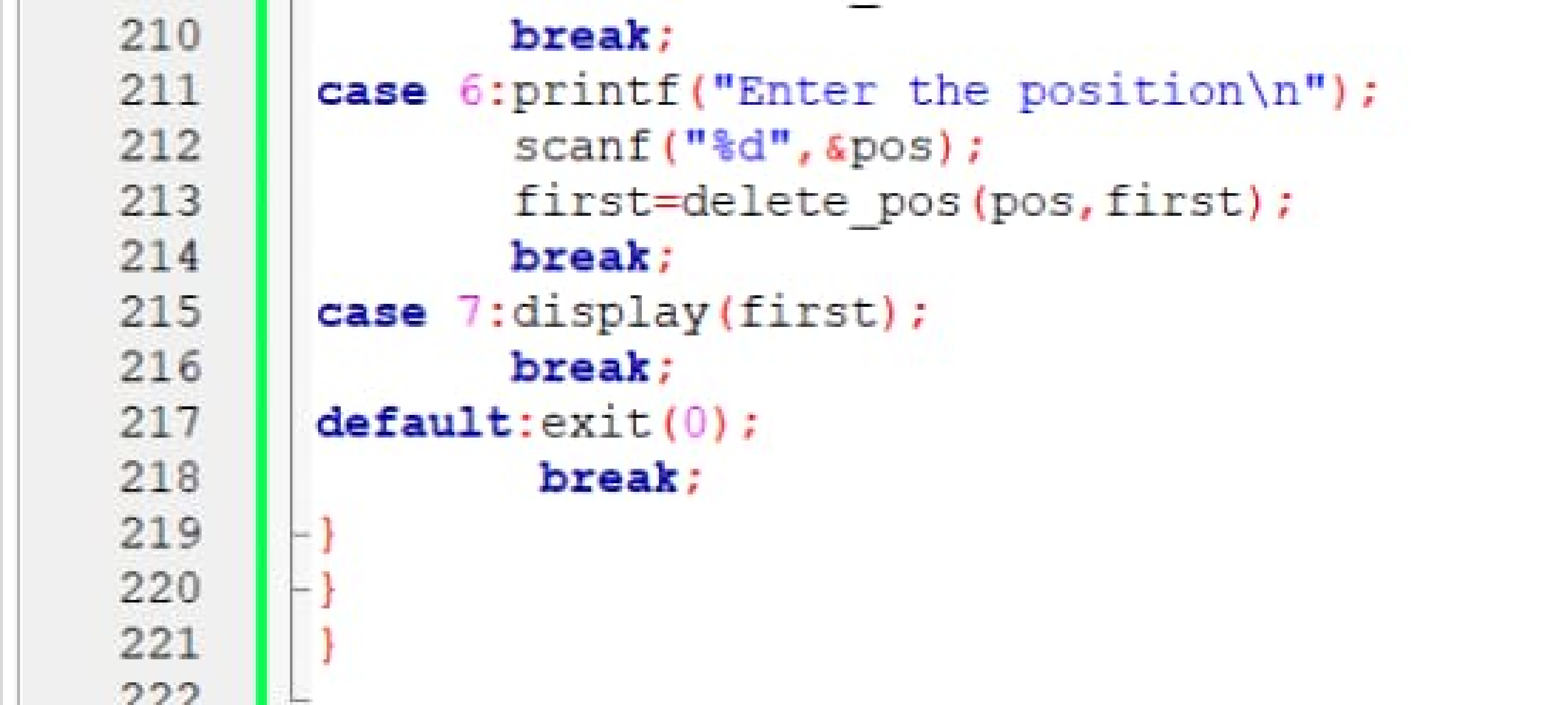
```c
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("ITEM DELETED AT REAR END=%d\n",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE delete_pos(int pos,NODE first)
{
NODE cur;
NODE prev;
int count,flag=0;
if(first==NULL || pos<0)
{
printf("invalid position\n");
return NULL;
}
if(pos==1)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
```



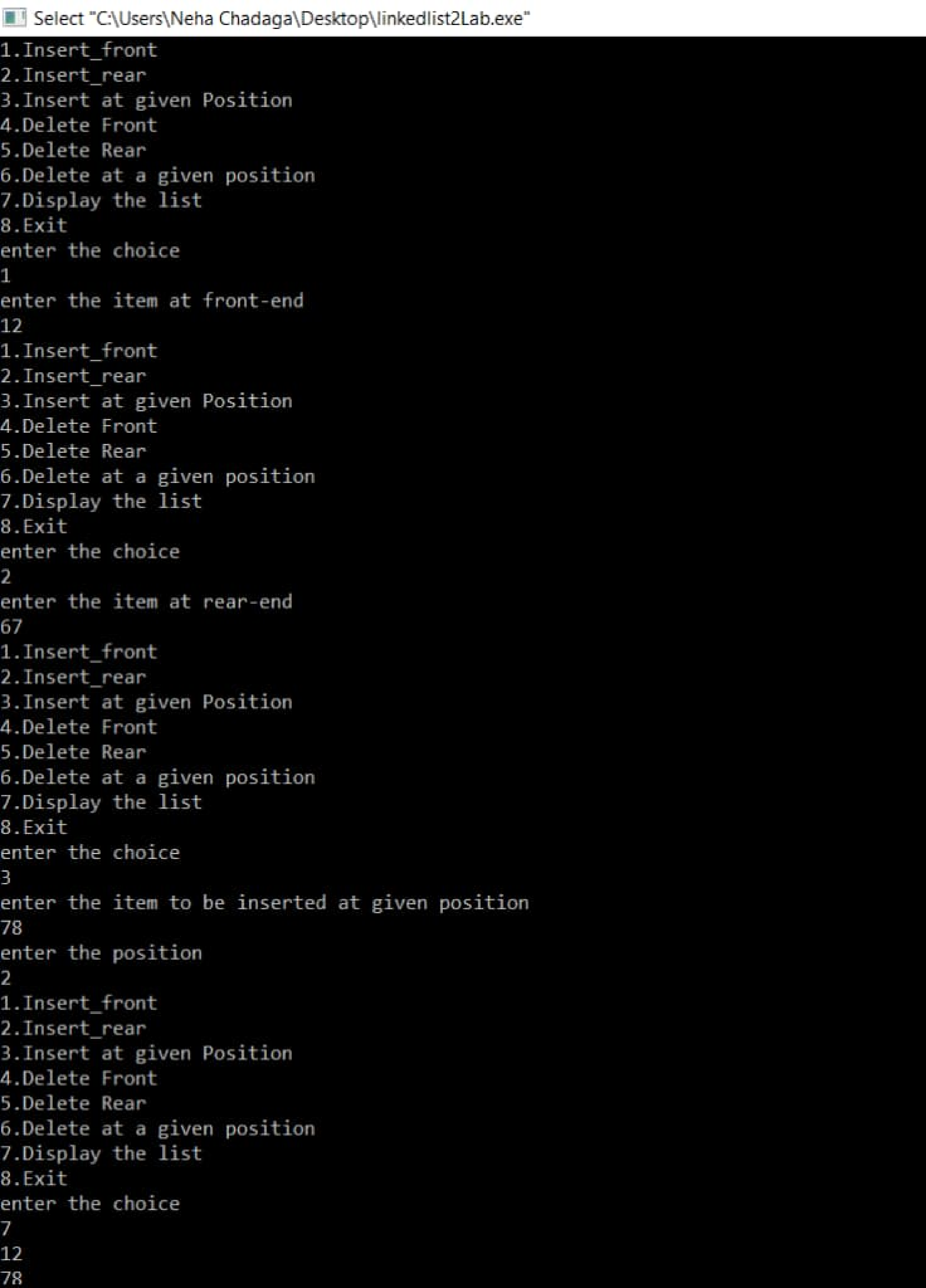
```c
while(cur!=NULL)
{
if(count==pos)
{
flag=1;
break;
}
count++;
prev=cur;
cur=cur->link;
}
if(flag==0)
{
printf("invalid position\n");
return first;
}
printf("ITEM DELETED AT POSITION %d is %d\n",pos,cur->info);
prev->link=cur->link;
freenode(cur);
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
```



```c
}
void main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("1.Insert front\n2.Insert rear\n3.Insert at given Position\n4.Delete Front\n5.Delete Rear\n6.Delete at a given position\n7.Display the list\n8.Exit\n"
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at front-end\n");
       scanf("%d",&item);
       first=insert_front(first,item);
       break;
case 2:printf("enter the item at rear-end\n");
       scanf("%d",&item);
       first=insert_rear(first,item);
       break;
case 3:printf("enter the item to be inserted at given position\n");
       scanf("%d",&item);
       printf("enter the position\n");
       scanf("%d",&pos);
       first=insert_pos(item,pos,first);
       break;
case 4:first=delete_front(first);
       break;
case 5:first=delete_rear(first);
       break;
```

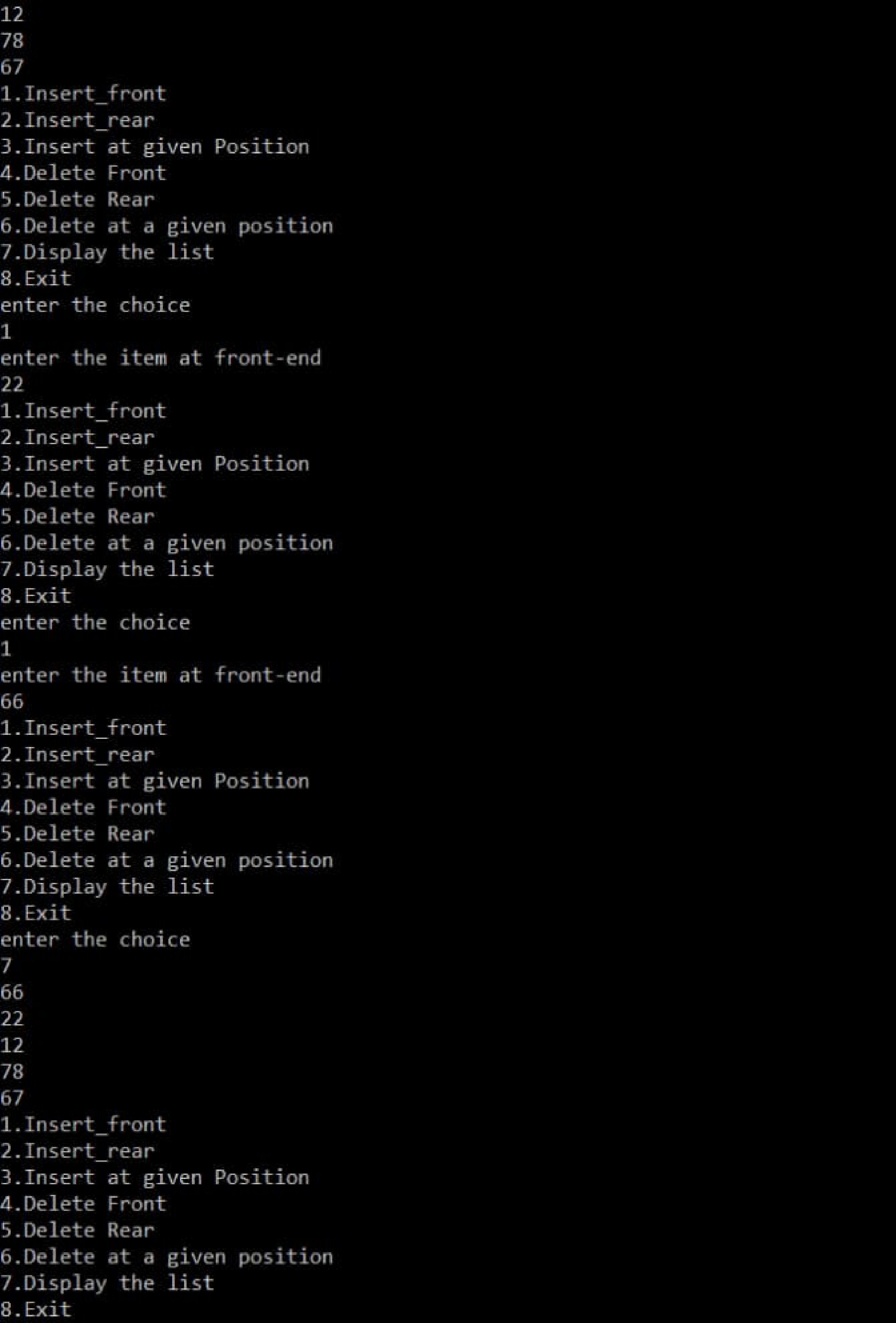


```c
       break;
case 6:printf("Enter the position\n");
       scanf("%d",&pos);
       first=delete_pos(pos,first);
       break;
case 7:display(first);
       break;
default:exit(0);
        break;
}
}
}
```

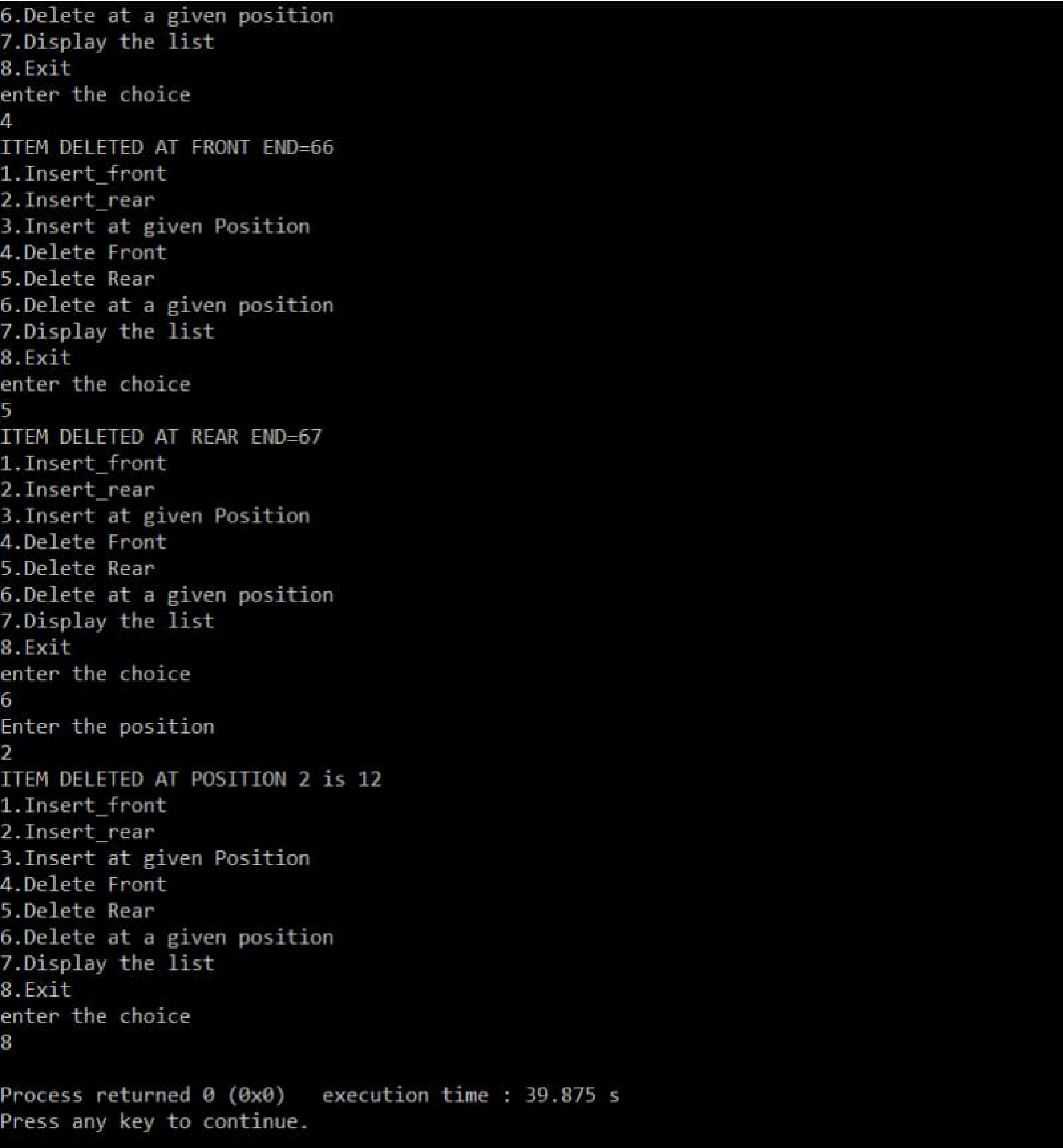


Select "C:\Users\Neha Chadaga\Desktop\linkedlist2Lab.exe"

```
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
12
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
2
enter the item at rear-end
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
3
enter the item to be inserted at given position
78
enter the position
2
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
7
12
78
```



```
12
78
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
22
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
1
enter the item at front-end
66
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
7
66
22
12
78
67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
```


```
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
4
ITEM DELETED AT FRONT END=66
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
5
ITEM DELETED AT REAR END=67
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
6
Enter the position
2
ITEM DELETED AT POSITION 2 is 12
1.Insert_front
2.Insert_rear
3.Insert at given Position
4.Delete Front
5.Delete Rear
6.Delete at a given position
7.Display the list
8.Exit
enter the choice
8

Process returned 0 (0x0)   execution time : 39.875 s
Press any key to continue.
```