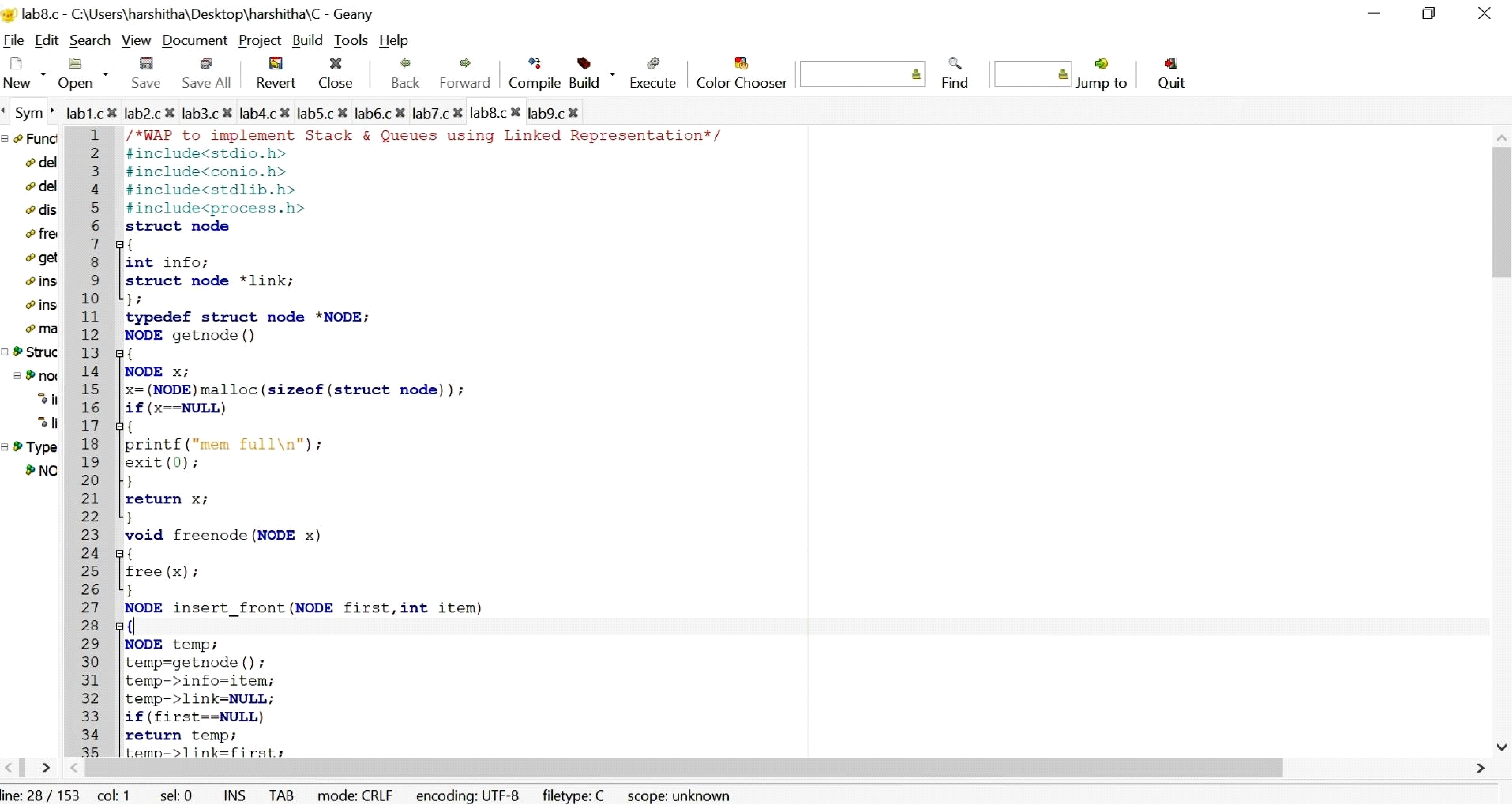
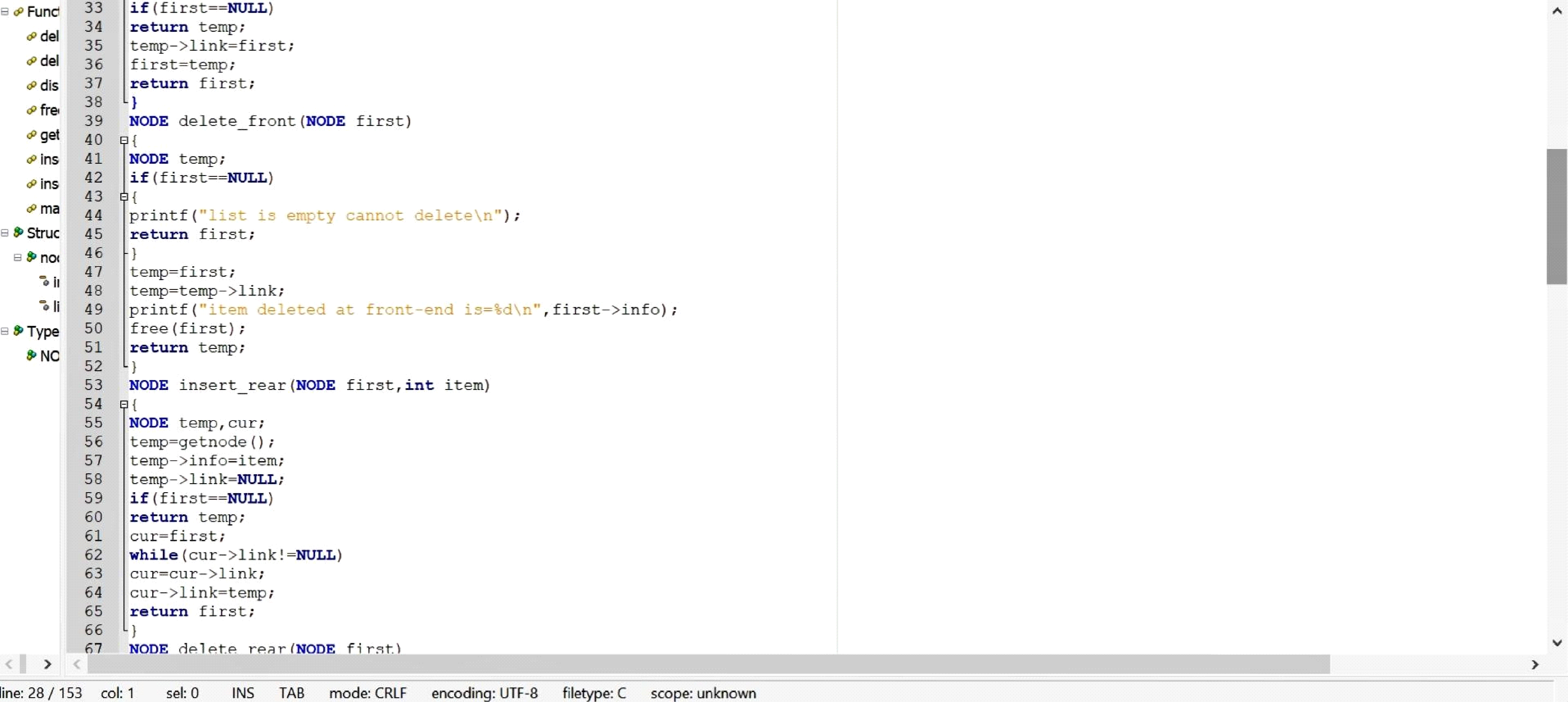

lab8.c - C:\Users\harshitha\Desktop\harshitha\C - Geany
File Edit Search View Document Project Build Tools Help
New Open Save Save All Revert Close Back Forward Compile Build Execute Color Chooser Find Jump to Quit
Sym lab1.c lab2.c lab3.c lab4.c lab5.c lab6.c lab7.c lab8.c lab9.c
Funct del del dis fre get ins ins ma Struc noc in li Type NO

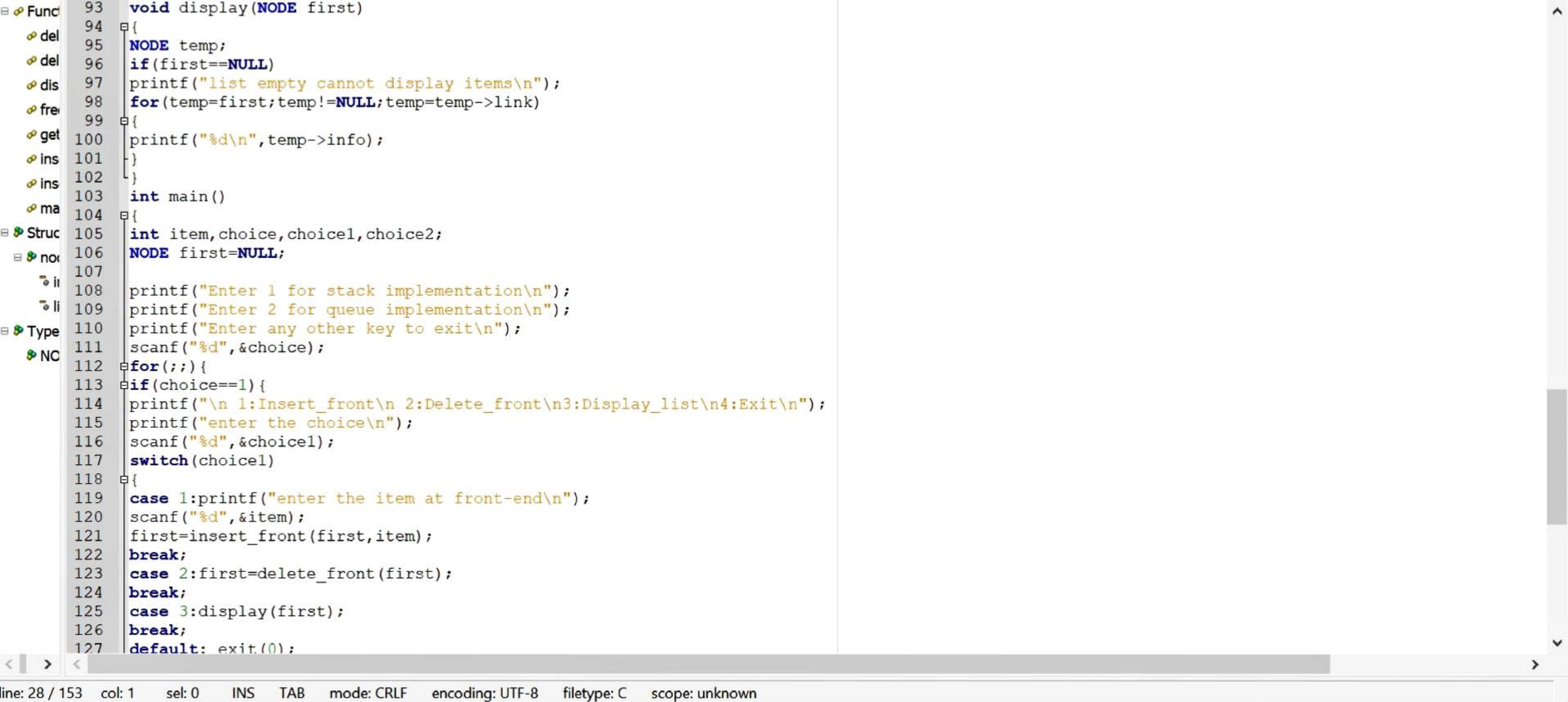
```c
/*WAP to implement Stack & Queues using Linked Representation*/
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>
struct node
{
int info;
struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
{
printf("mem 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)
return temp;
temp->link=first;
```

line: 28 / 153 col: 1 sel: 0 INS TAB mode: CRLF encoding: UTF-8 filetype: C scope: unknown



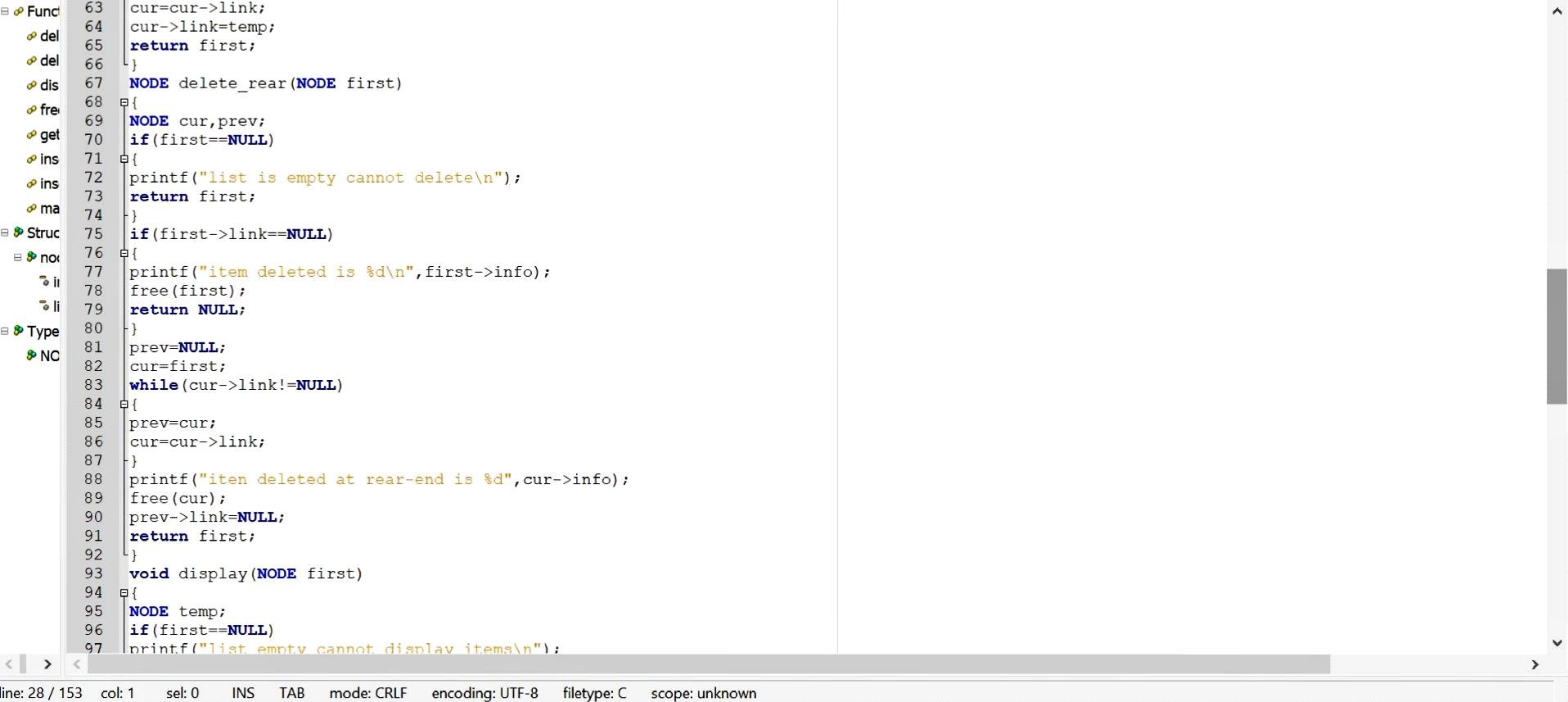
```
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}
NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
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 delete_rear(NODE first)
```

line: 28 / 153    col: 1    sel: 0    INS    TAB    mode: CRLF    encoding: UTF-8    filetype: C    scope: unknown



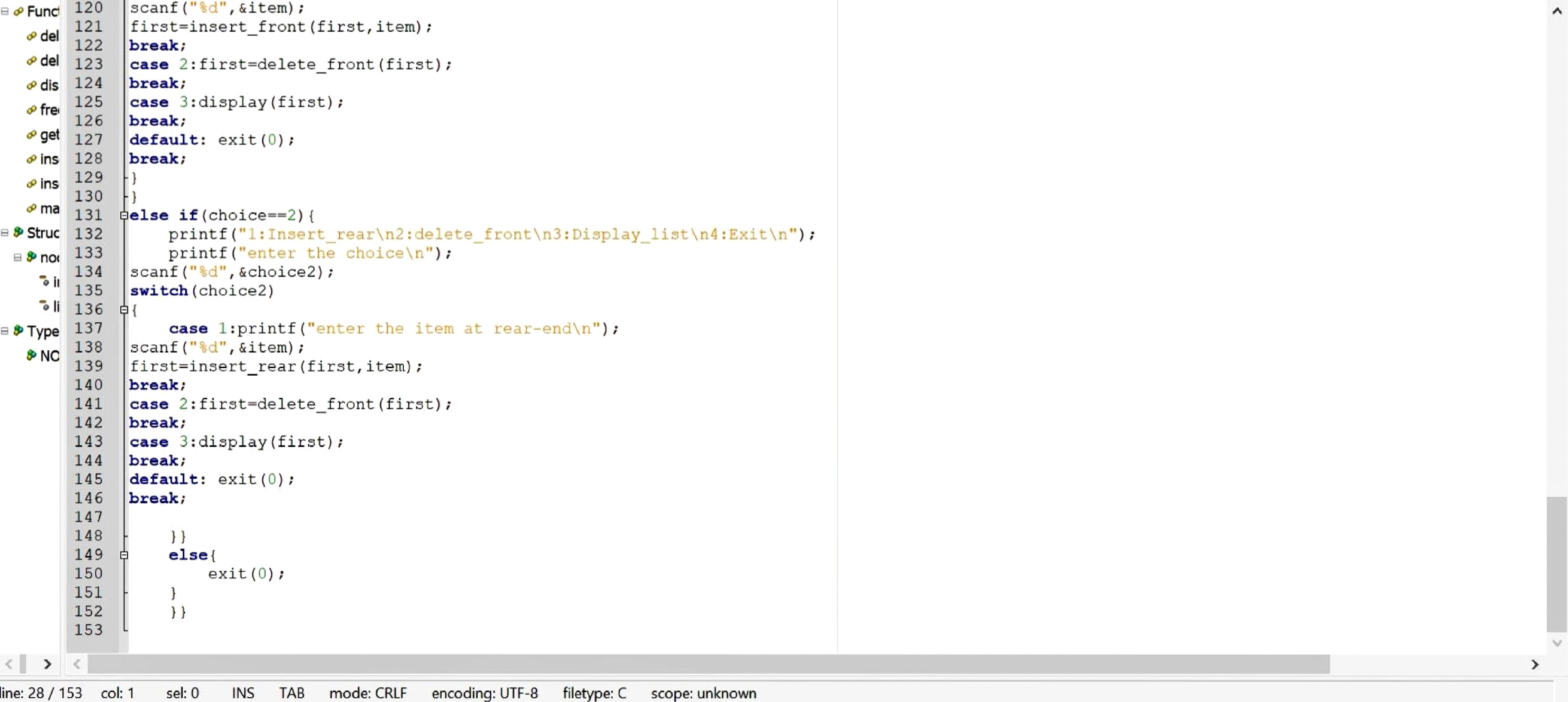
```c
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
for(temp=first;temp!=NULL;temp=temp->link)
{
printf("%d\n",temp->info);
}
}
int main()
{
int item,choice,choice1,choice2;
NODE first=NULL;

printf("Enter 1 for stack implementation\n");
printf("Enter 2 for queue implementation\n");
printf("Enter any other key to exit\n");
scanf("%d",&choice);
for(;;){
if(choice==1){
printf("\n 1:Insert_front\n 2:Delete_front\n3:Display_list\n4:Exit\n");
printf("enter the choice\n");
scanf("%d",&choice1);
switch(choice1)
{
case 1:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default: exit(0);
```

line: 28 / 153    col: 1    sel: 0    INS    TAB    mode: CRLF    encoding: UTF-8    filetype: C    scope: unknown



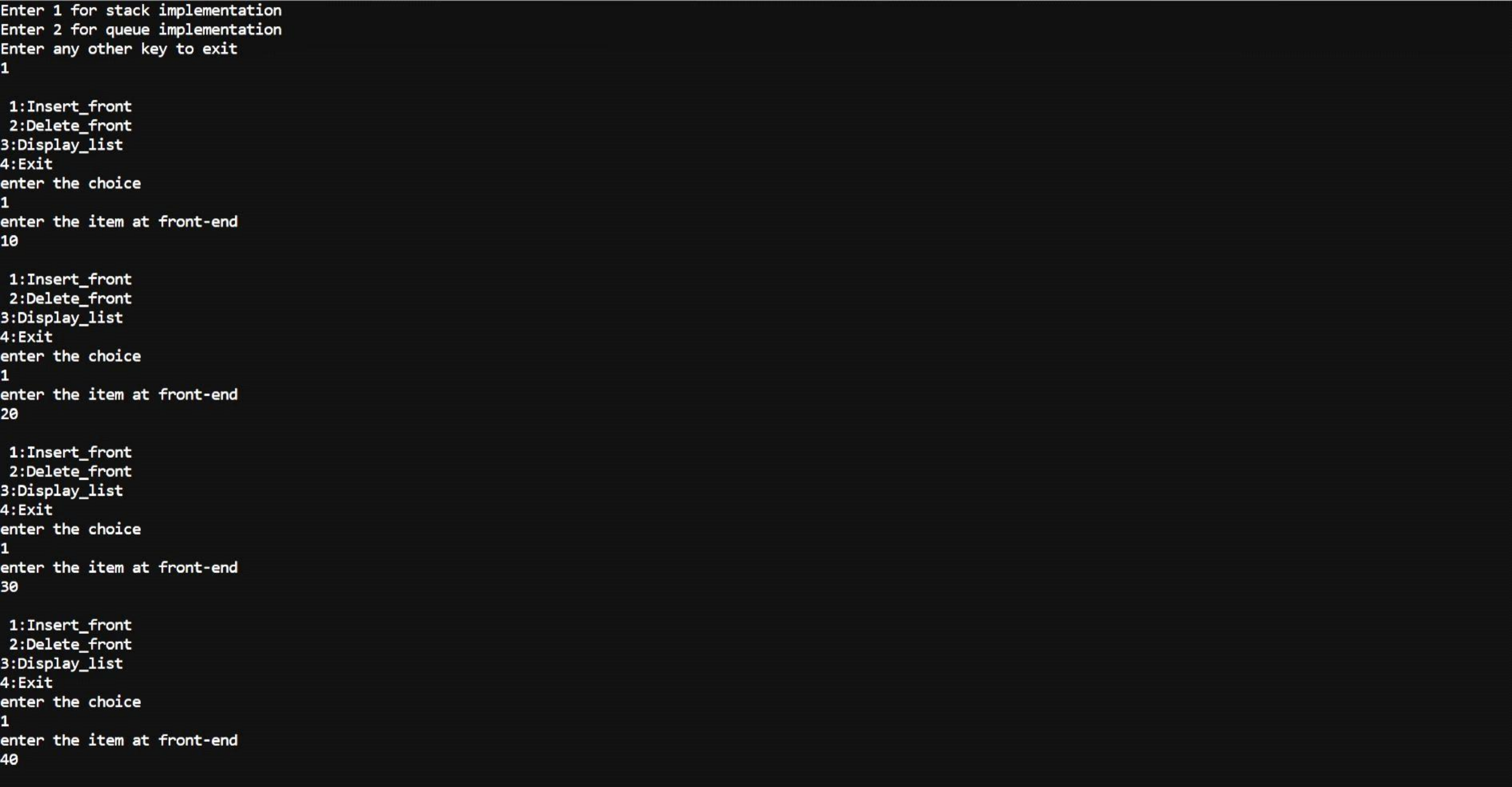
```
cur=cur->link;
cur->link=temp;
return first;
}
NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("iten deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
void display(NODE first)
{
NODE temp;
if(first==NULL)
printf("list empty cannot display items\n");
```

line: 28 / 153 col: 1 sel: 0 INS TAB mode: CRLF encoding: UTF-8 filetype: C scope: unknown

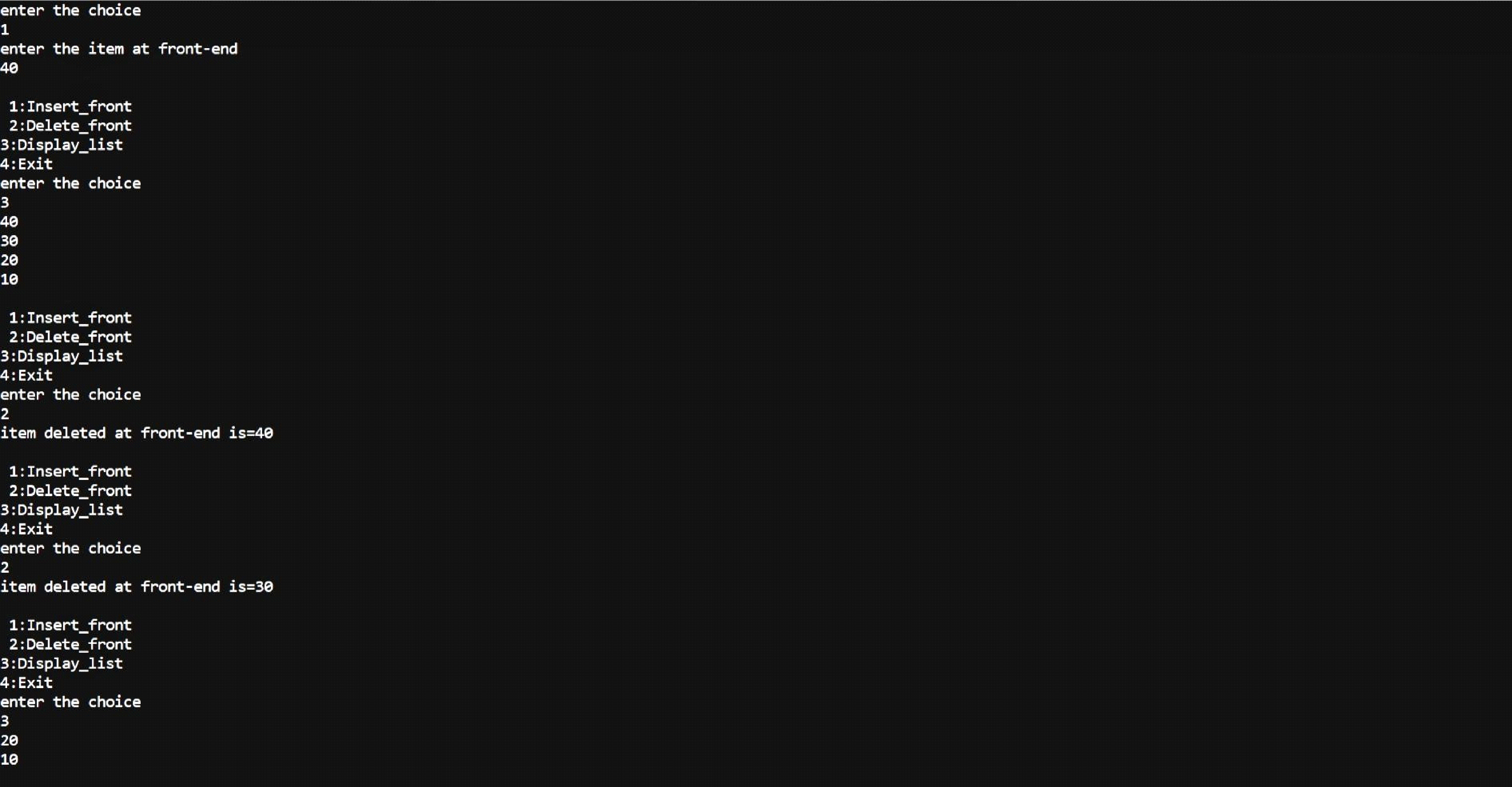


```c
scanf("%d",&item);
first=insert_front(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default: exit(0);
break;
}
}
else if(choice==2){
    printf("1:Insert_rear\n2:delete_front\n3:Display_list\n4:Exit\n");
    printf("enter the choice\n");
scanf("%d",&choice2);
switch(choice2)
{
    case 1:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
default: exit(0);
break;

    }}
    else{
        exit(0);
    }
    }}
```

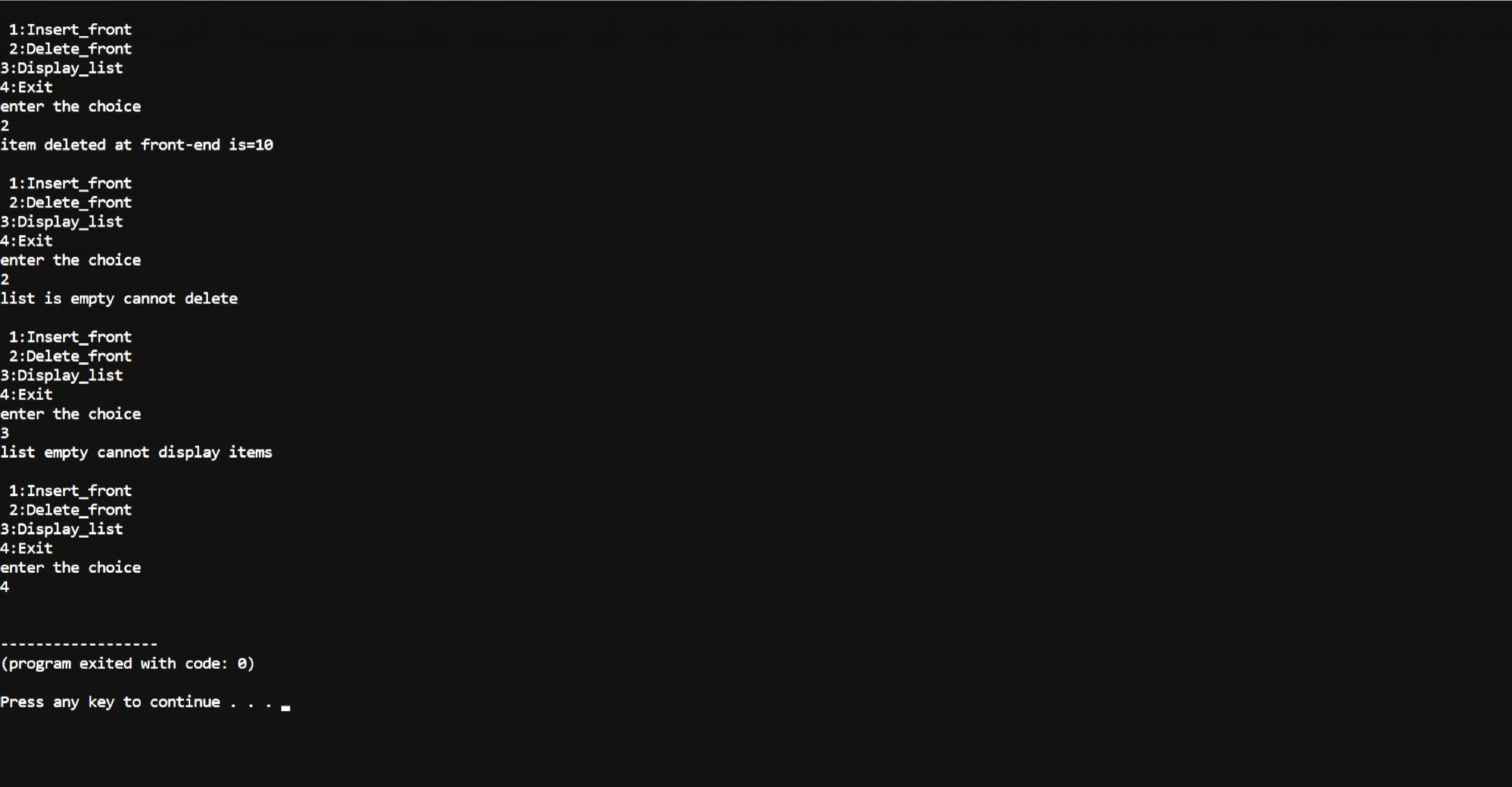
```
line: 28 / 153   col: 1   sel: 0   INS   TAB   mode: CRLF   encoding: UTF-8   filetype: C   scope: unknown
```



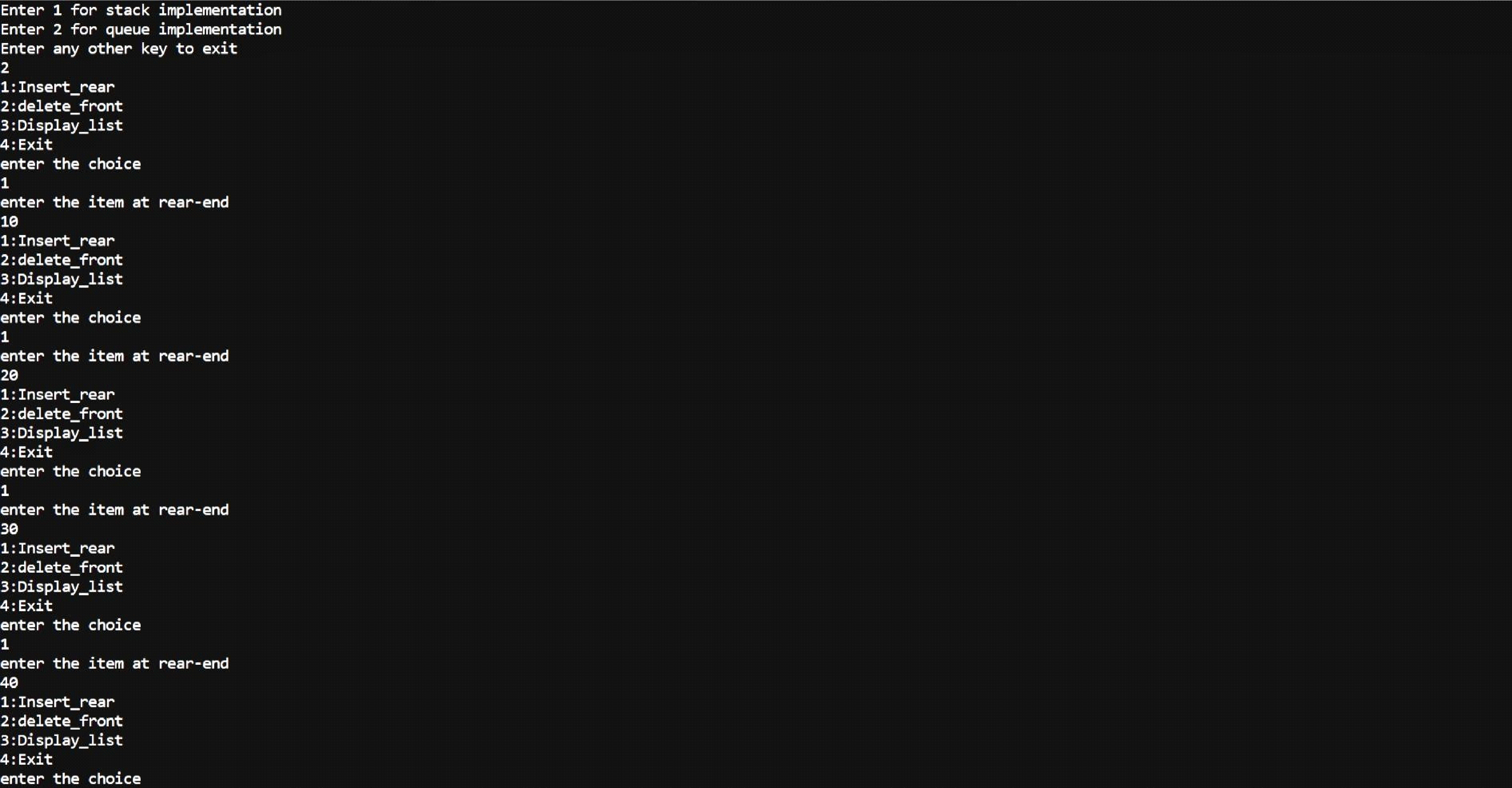
```
Enter 1 for stack implementation
Enter 2 for queue implementation
Enter any other key to exit
1

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
20

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
30

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
40
```



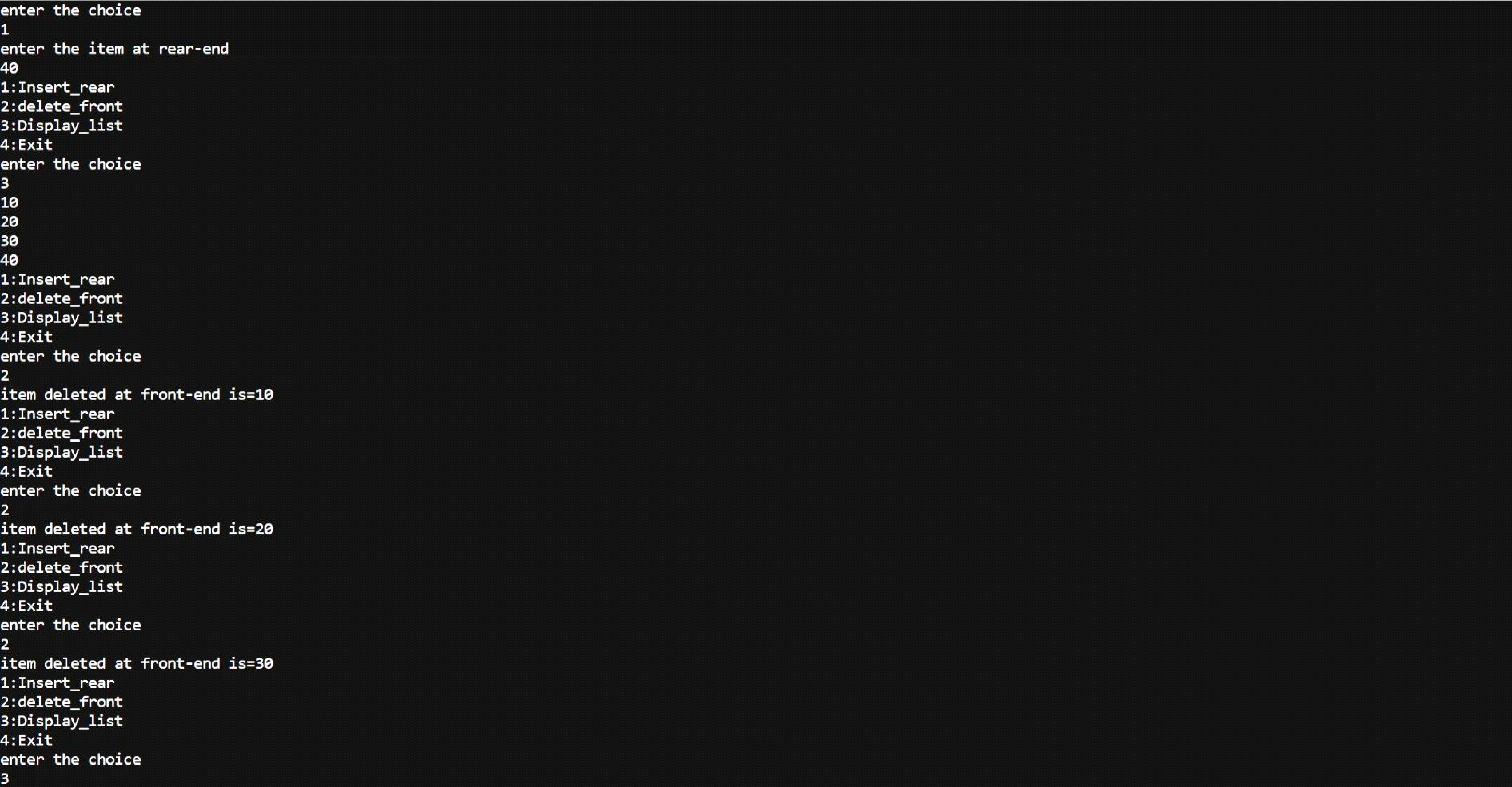
```
enter the choice
1
enter the item at front-end
40

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
40
30
20
10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=40

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
20
10
```



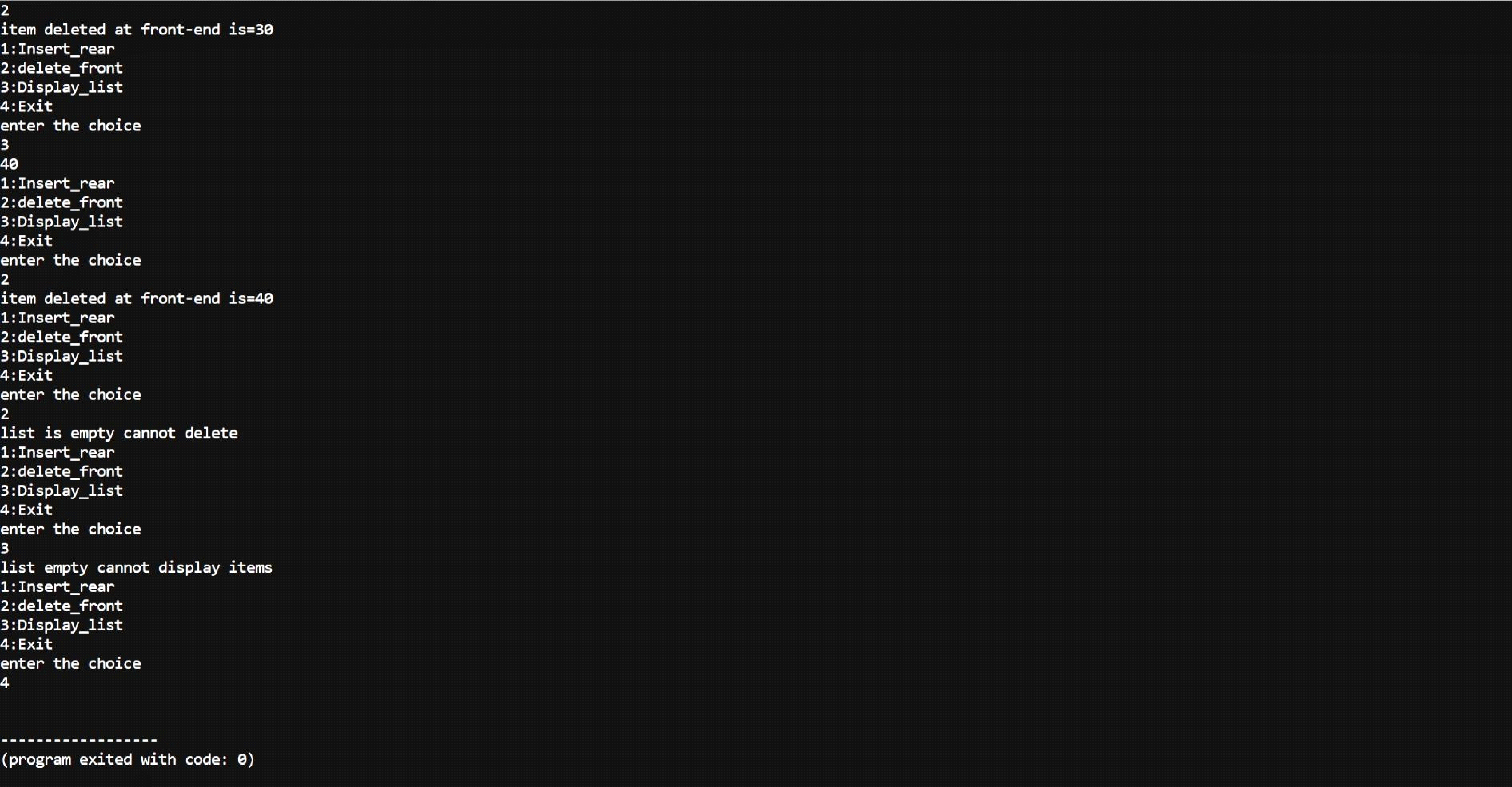
```
 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
2
list is empty cannot delete

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
3
list empty cannot display items

 1:Insert_front
 2:Delete_front
3:Display_list
4:Exit
enter the choice
4



------------------
(program exited with code: 0)

Press any key to continue . . .
```



```
Enter 1 for stack implementation
Enter 2 for queue implementation
Enter any other key to exit
2
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
10
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
20
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
30
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
```



```
enter the choice
1
enter the item at rear-end
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
10
20
30
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=10
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=20
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
```



```
2
item deleted at front-end is=30
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=40
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
2
list is empty cannot delete
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
3
list empty cannot display items
1:Insert_rear
2:delete_front
3:Display_list
4:Exit
enter the choice
4



------------------
(program exited with code: 0)
```