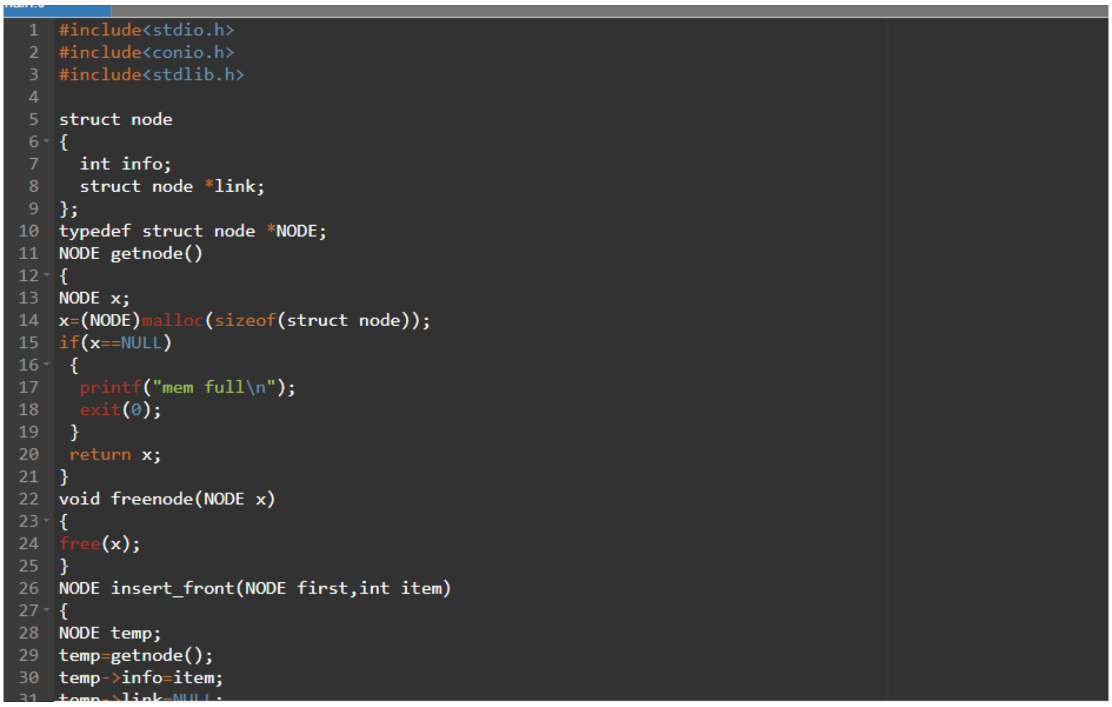
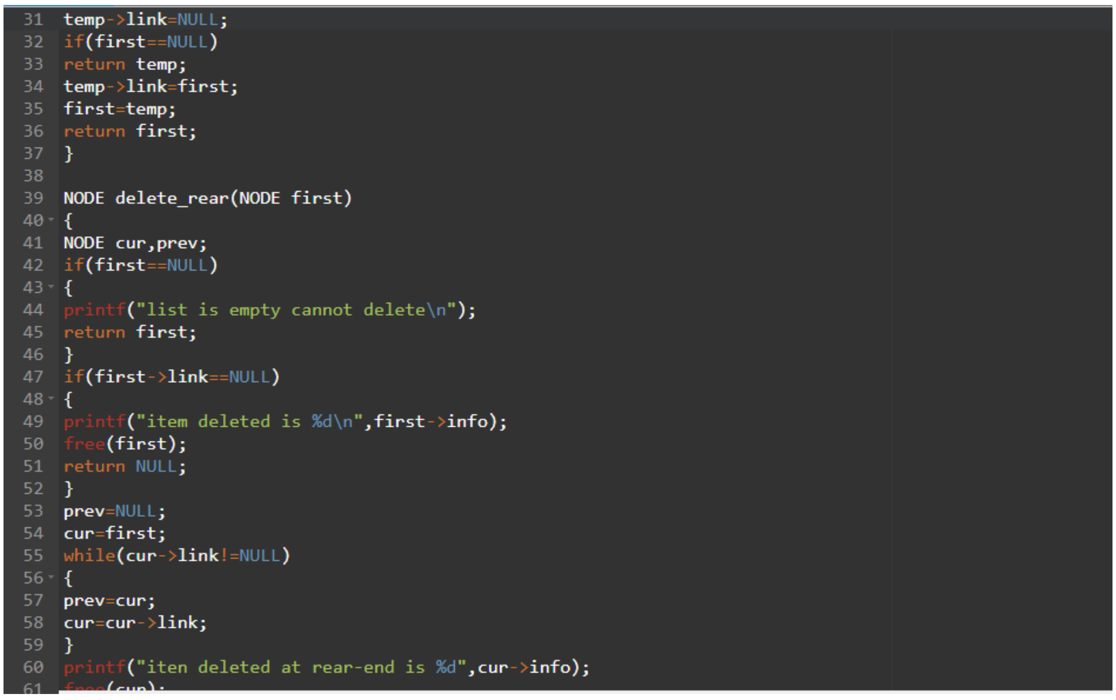
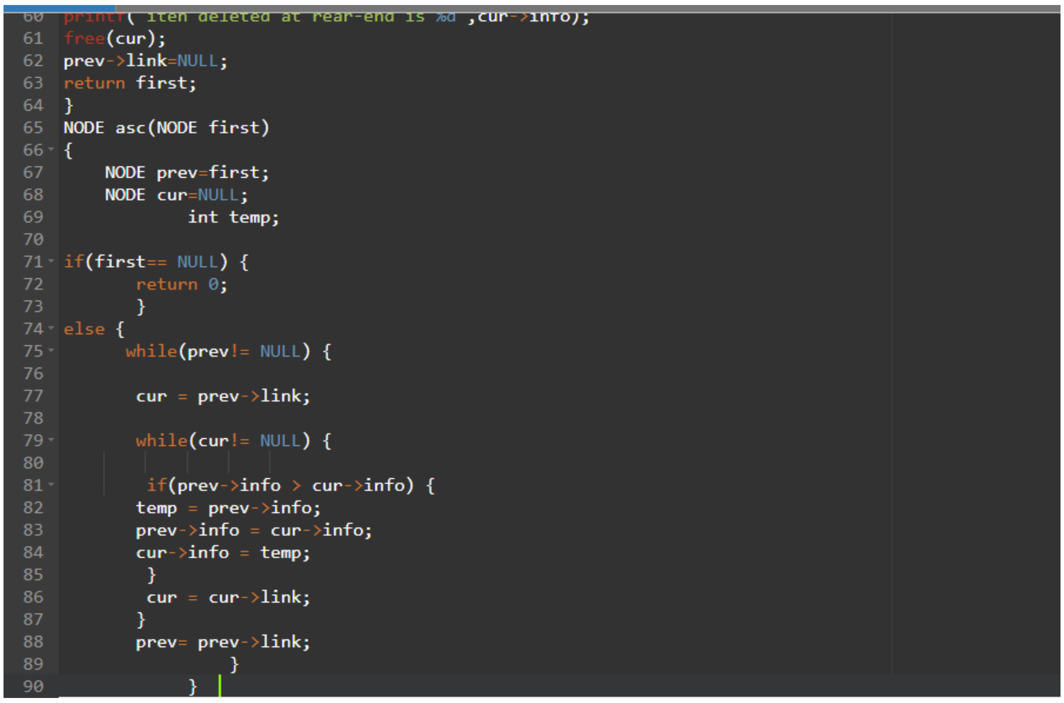
PRACTICE PROG:



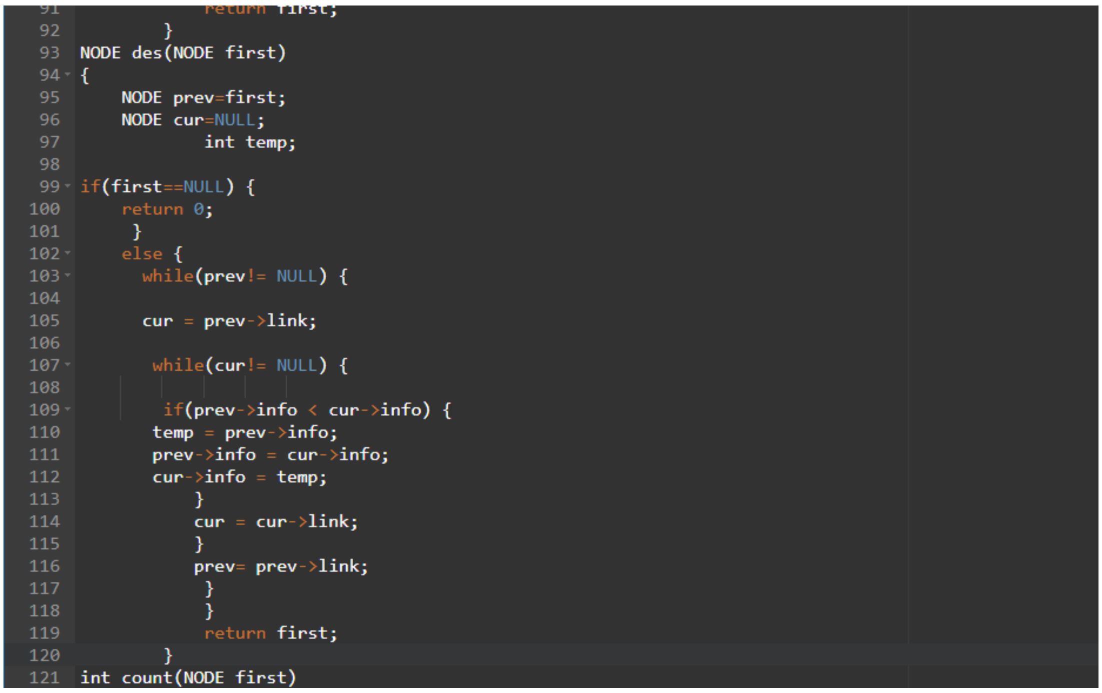
```c
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

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;
```



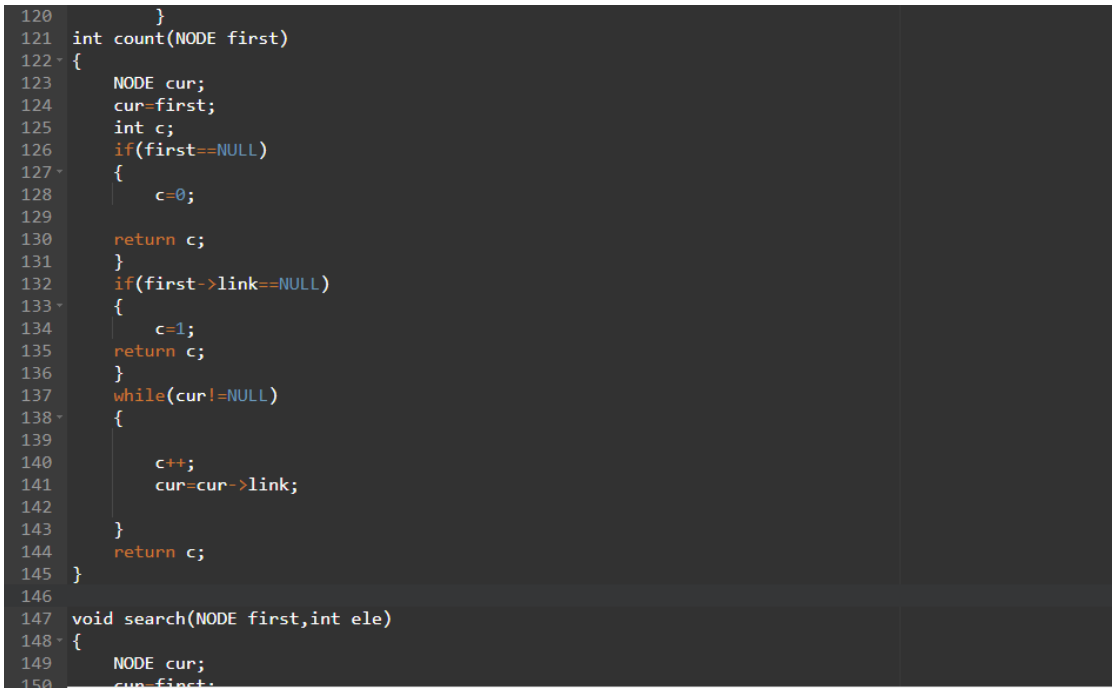
```c
temp->link=NULL;
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}

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);
```



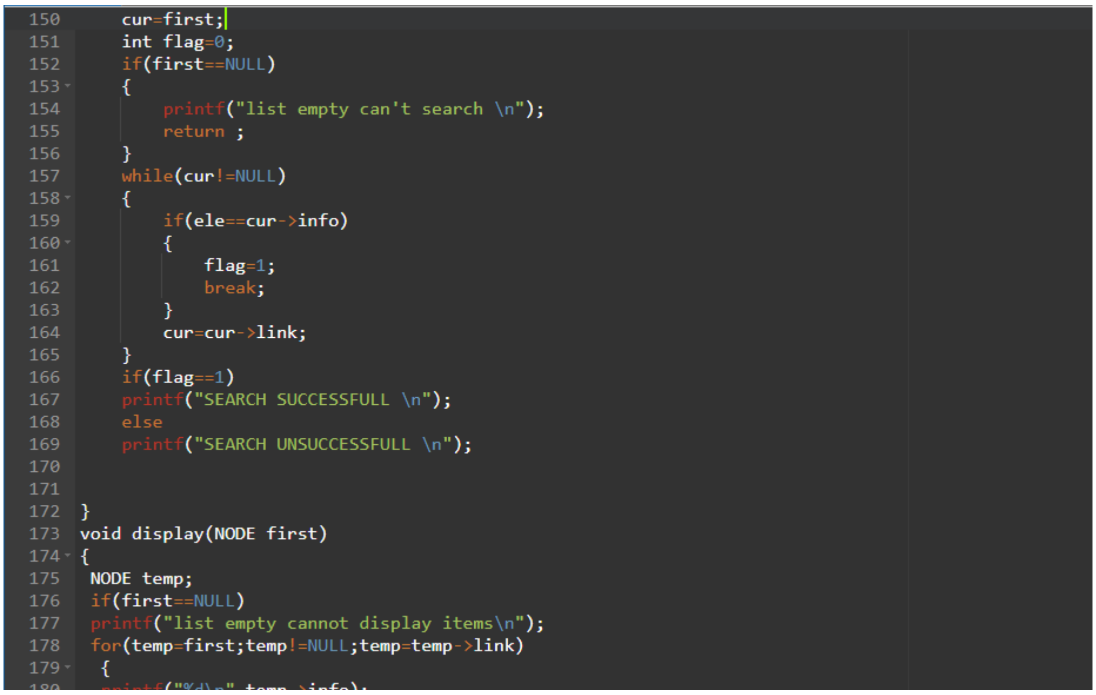
```
printf("item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
NODE asc(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
            int temp;

if(first== NULL) {
        return 0;
        }
else {
       while(prev!= NULL) {

        cur = prev->link;

        while(cur!= NULL) {

         if(prev->info > cur->info) {
        temp = prev->info;
        prev->info = cur->info;
        cur->info = temp;
         }
         cur = cur->link;
        }
        prev= prev->link;
               }
         }
```



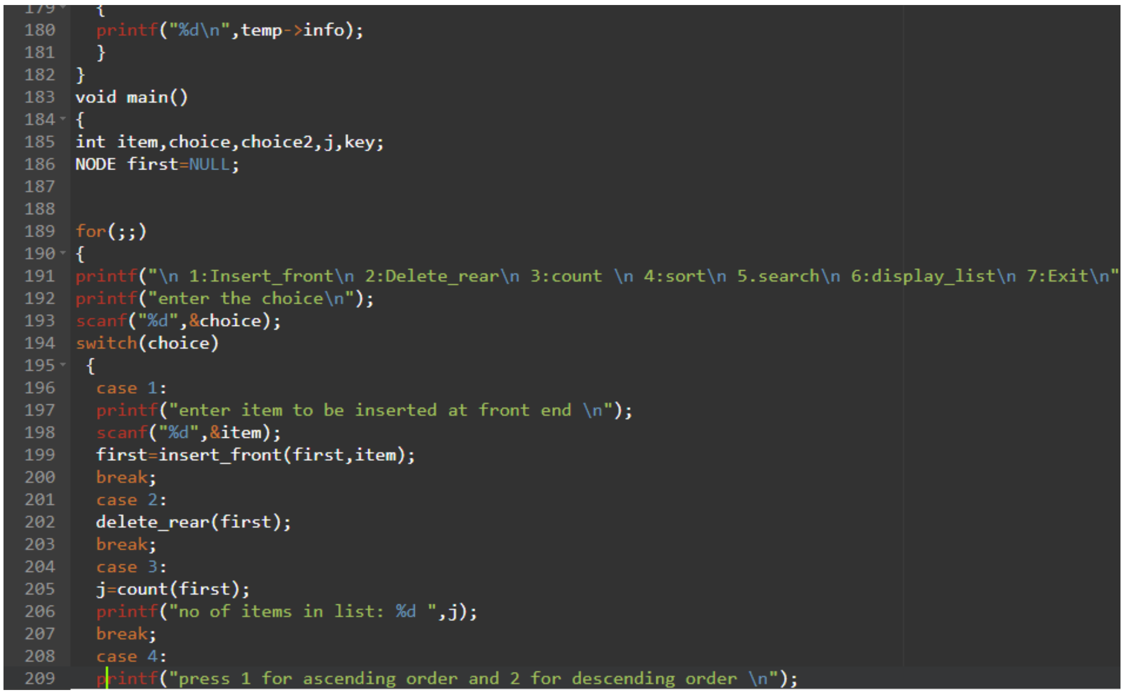
```
            return first;
        }
NODE des(NODE first)
{
    NODE prev=first;
    NODE cur=NULL;
            int temp;

if(first==NULL) {
    return 0;
     }
    else {
      while(prev!= NULL) {

      cur = prev->link;

       while(cur!= NULL) {

        if(prev->info < cur->info) {
       temp = prev->info;
       prev->info = cur->info;
       cur->info = temp;
           }
           cur = cur->link;
           }
           prev= prev->link;
            }
            }
            return first;
        }
int count(NODE first)
```



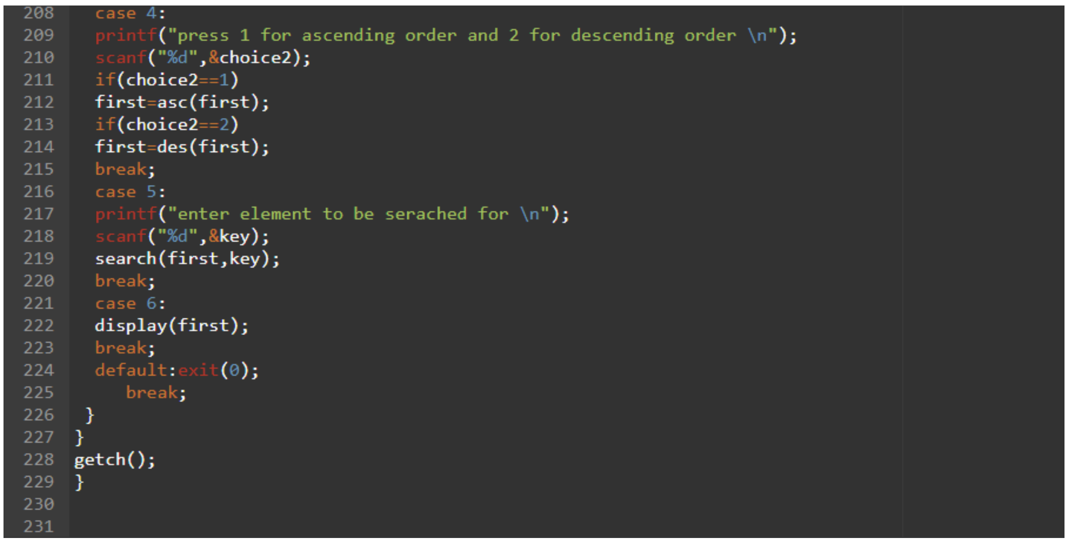
```c
        }
int count(NODE first)
{
    NODE cur;
    cur=first;
    int c;
    if(first==NULL)
    {
        c=0;

    return c;
    }
    if(first->link==NULL)
    {
        c=1;
    return c;
    }
    while(cur!=NULL)
    {

        c++;
        cur=cur->link;

    }
    return c;
}

void search(NODE first,int ele)
{
    NODE cur;
```



```c
    cur=first;
    int flag=0;
    if(first==NULL)
    {
        printf("list empty can't search \n");
        return ;
    }
    while(cur!=NULL)
    {
        if(ele==cur->info)
        {
            flag=1;
            break;
        }
        cur=cur->link;
    }
    if(flag==1)
    printf("SEARCH SUCCESSFULL \n");
    else
    printf("SEARCH UNSUCCESSFULL \n");


}
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);
```

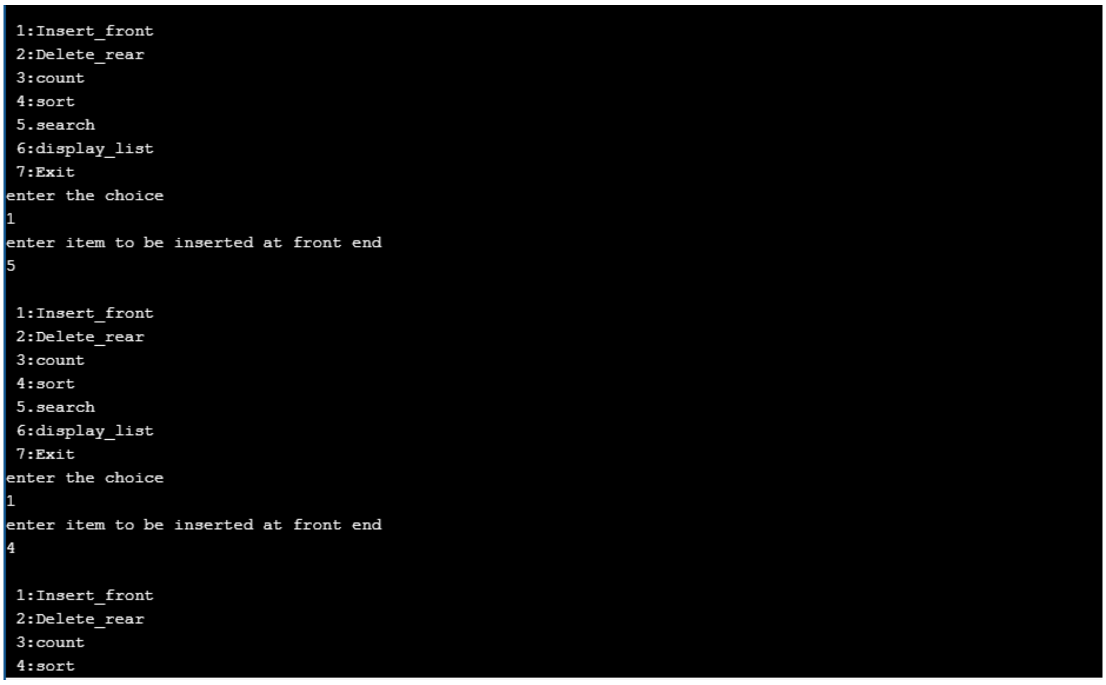


```c
  {
  printf("%d\n",temp->info);
  }
}
void main()
{
int item,choice,choice2,j,key;
NODE first=NULL;


for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_rear\n 3:count \n 4:sort\n 5.search\n 6:display_list\n 7:Exit\n"
printf("enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:
  printf("enter item to be inserted at front end \n");
  scanf("%d",&item);
  first=insert_front(first,item);
  break;
  case 2:
  delete_rear(first);
  break;
  case 3:
  j=count(first);
  printf("no of items in list: %d ",j);
  break;
  case 4:
  printf("press 1 for ascending order and 2 for descending order \n");
```

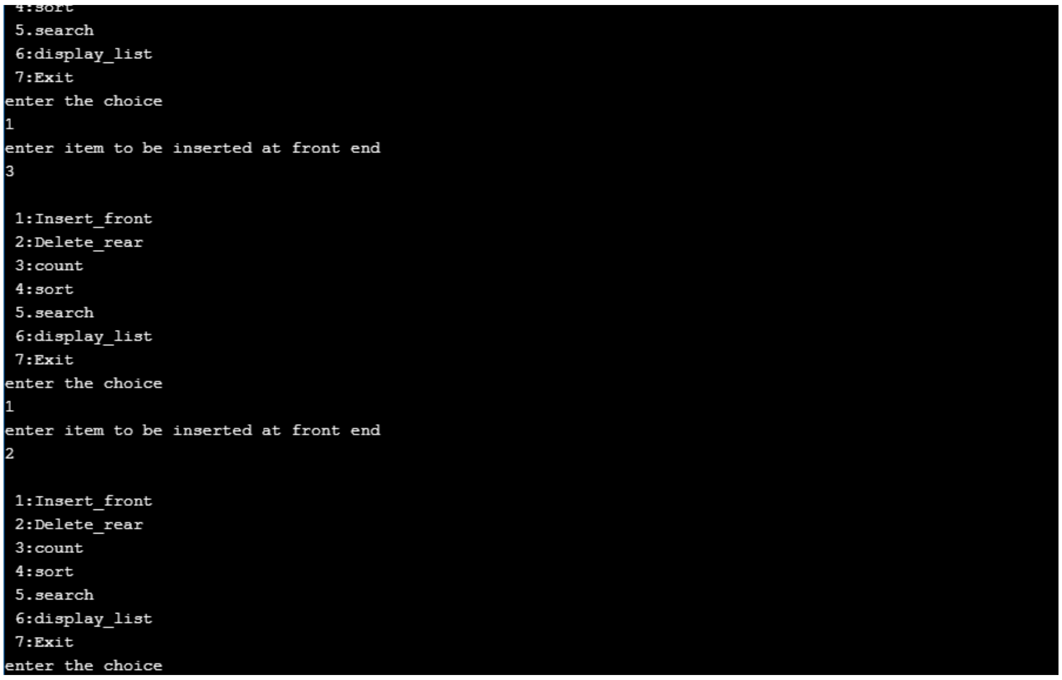


```c
  case 4:
  printf("press 1 for ascending order and 2 for descending order \n");
  scanf("%d",&choice2);
  if(choice2==1)
  first=asc(first);
  if(choice2==2)
  first=des(first);
  break;
  case 5:
  printf("enter element to be serached for \n");
  scanf("%d",&key);
  search(first,key);
  break;
  case 6:
  display(first);
  break;
  default:exit(0);
     break;
 }
}
getch();
}
```

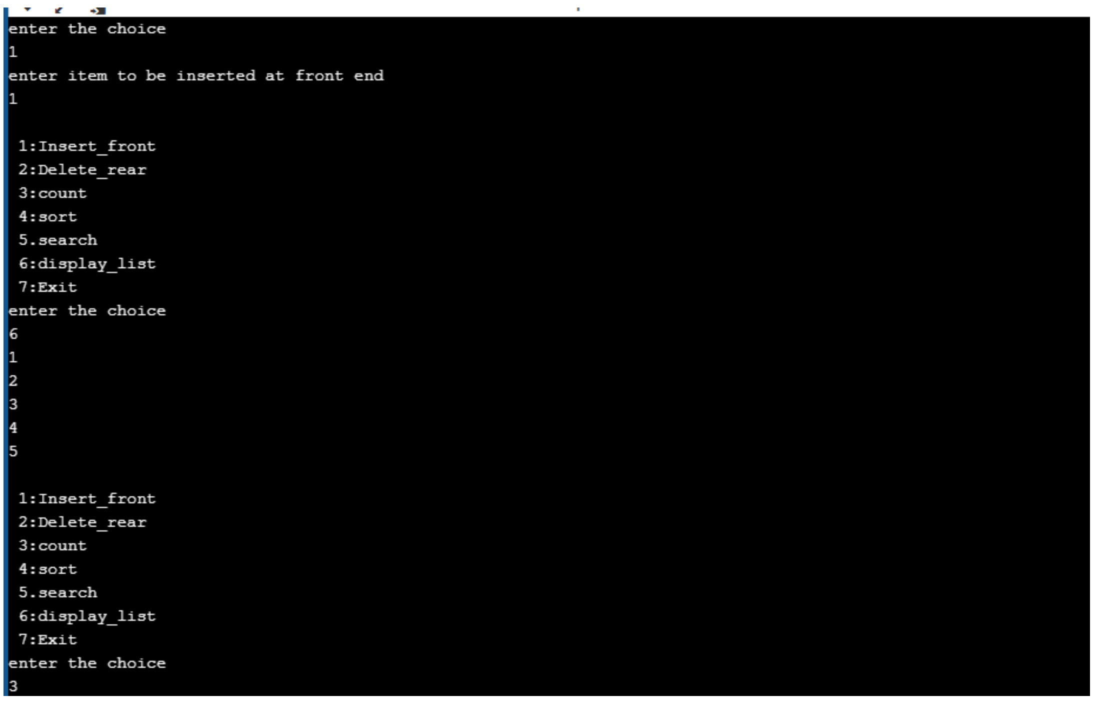
OUTPUT:



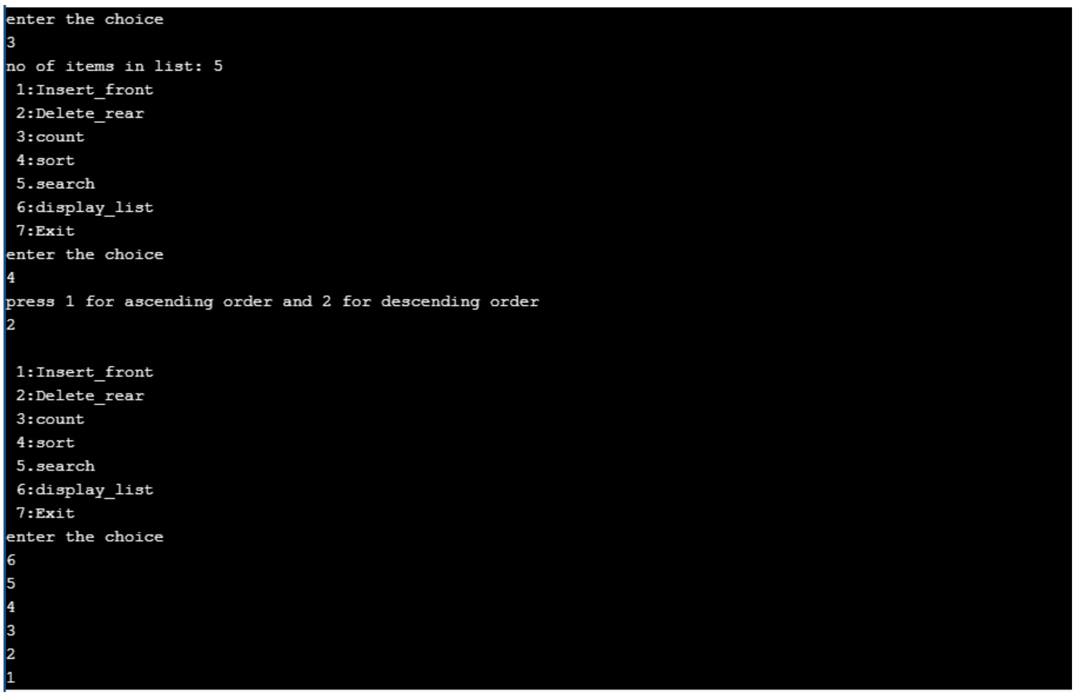
```
 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
1
enter item to be inserted at front end
5

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
1
enter item to be inserted at front end
4

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
```



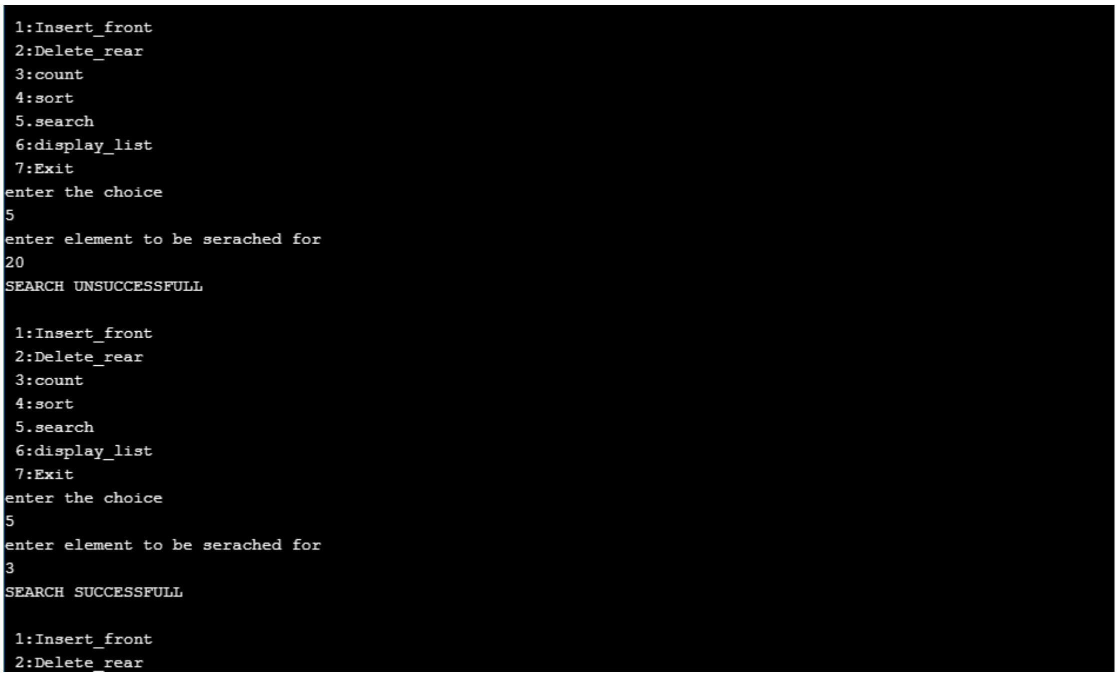
```
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
1
enter item to be inserted at front end
3

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
1
enter item to be inserted at front end
2

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
```



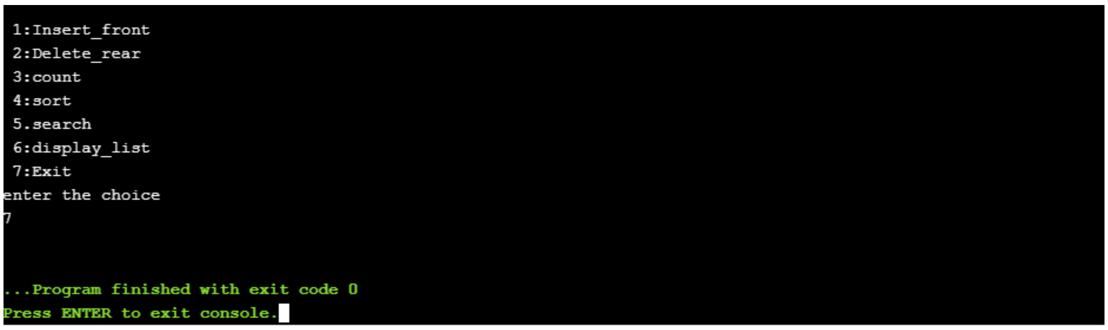
```
enter the choice
1
enter item to be inserted at front end
1

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
6
1
2
3
4
5

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
3
```



```
enter the choice
3
no of items in list: 5
 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
4
press 1 for ascending order and 2 for descending order
2

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
6
5
4
3
2
1
```



```
 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
5
enter element to be serached for
20
SEARCH UNSUCCESSFULL

 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
5
enter element to be serached for
3
SEARCH SUCCESSFULL

 1:Insert_front
 2:Delete_rear
```



```
 1:Insert_front
 2:Delete_rear
 3:count
 4:sort
 5.search
 6:display_list
 7:Exit
enter the choice
7


...Program finished with exit code 0
Press ENTER to exit console.
```