```
#include<stdio.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;
temp->link=NULL;
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)
{
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("item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}
```

```
NODE order_list(int item,NODE first)
{
NODE temp,prev,cur;
temp=getnode();
temp->info=item;
```

```
temp->link=NULL;
if(first==NULL) return temp;
if(item<first->info)
{
temp->link=first;
return temp;
}
prev=NULL;
cur=first;
while(cur!=NULL&&item>cur->info)
{
prev=cur;
cur=cur->link;
}
prev->link=temp;
temp->link=cur;
return first;
}
```

```
NODE insert_at_pos(int pos,NODE first,int item){

    NODE cur,prev,temp;

    cur = first ;

    int c=1;
```

```
temp = getnode();
temp->info=item;
temp->link=NULL;

if(pos==0){
    printf("Invalid position");
    return first;
}

if(first==NULL&&pos==1){
    first=temp;
    return first;
}

if(pos==1){
    temp->link = cur;
    first = temp;
    return first;
}

prev = NULL;


while(cur!=NULL&&c<pos){
    prev = cur;
    cur = cur->link;
    c++;
}
```

```
    if(cur==NULL){
        printf("Invalid POS\n");
    }

    prev->link = temp;
    temp->link = cur;
    return first;
}



NODE delete_at_pos(int pos,NODE first){

    NODE cur,prev;
    int c=1;

    if(pos==0){
            printf("Invalid POS\n");
            return first;
}
  if(first==NULL){
    printf("No elements\n");
    return first;
  }

     if(first->link==NULL&&pos==1){
        free(first);
        return NULL;
     }
```

```
    cur=first;

    while(cur!=NULL&&c<pos){
       prev =cur;
       cur = cur->link;
       c++;
    }
    if(cur==NULL){
       printf("Invalid pos\n");
       return first;
    }

    prev->link = cur->link;
    free(cur);
    return first;
}
```

```
int length(NODE first)
{
NODE cur;
int count=0;
if(first==NULL) return 0;
cur=first;
while(cur!=NULL)
{
count++;
cur=cur->link;
```

```
}
return count;
}
```

```
NODE concat(NODE first,NODE second)
{
 NODE cur;
 if(first==NULL)
  return second;
 if(second==NULL)
  return first;
 cur=first;
 while(cur->link!=NULL)
  cur=cur->link;
 cur->link=second;
 return first;
}
```

```
NODE reverse(NODE first)
 {
 NODE cur,temp;
 cur=NULL;
```

```
 while(first!=NULL)
  {
   temp=first;
   first=first->link;
   temp->link=cur;
   cur=temp;
  }
 return cur;
}




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,pos;
NODE first=NULL;
int n,i;
```

```
NODE a,b;



for(;;)

{

printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n
4:Delete_rear\n");

printf(" 5:Sort_list\n 6:insert_at_pos \n 7:delete_at_pos \n
8:Display_list\n 9:Concate\n 10.Reverse\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:first=delete_front(first);

       break;


  case 3:printf("enter the item at rear-end\n");

       scanf("%d",&item);

       first=insert_rear(first,item);

       break;


  case 4:first=delete_rear(first);

       break;


  case 5:printf("enter the item to be inserted in ordered_list\n");

       scanf("%d",&item);
```

```
     first=order_list(item,first);
     break;

case 6:printf("Enter pos to be inserted \n");
scanf("%d",&pos);
printf("Enter item\n");
scanf("%d",&item);

first = insert_at_pos(pos,first,item);
break;

case 7:
  printf("enter the pos to be deleted\n");
     scanf("%d",&pos);
     first=delete_at_pos(pos,first);
     break;

case 8:display(first);
     break;

case 9:

    printf("enter the no of nodes in 1\n");
          scanf("%d",&n);
          a=NULL;
          for(i=0;i<n;i++)
           {
            printf("enter the item\n");
            scanf("%d",&item);
            a=insert_rear(a,item);
```

```
            }
            printf("enter the no of nodes in 2\n");
           scanf("%d",&n);
           b=NULL;
           for(i=0;i<n;i++)
            {
             printf("enter the item\n");
             scanf("%d",&item);
             b=insert_rear(b,item);
            }
            a=concat(a,b);
            display(a);
           break;

   case 10:first=reverse(first);
           display(first);
           break;

  default:exit(0);
       break;
 }
}
```

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
3
enter the item at rear-end
8

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
3
enter the item at rear-end
9

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
3
enter the item at rear-end
4

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
3
enter the item at rear-end
5

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
5
enter the item to be inserted in ordered_list
10

 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Sort_list
 6:insert_at_pos
 7:delete_at_pos
 8:Display_list
 9:Concate
 10.Reverse
enter the choice
8
8
9
4
}
```