→ linked list program to count, sort and search.

```
#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("memory is full\n");
exit(0);
}
return x;
}
void freenode(NODE x)
{
free(x);
NODE insertfront(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 deleterear(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;
}
void display (NODE first)
{
NODE temp;
if(first ==NULL)
printf("List is empty. \n");
for(temp= first; temp!= NULL; temp= temp->link)
{
printf("%d \n", temp-> info );
}}
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;
}
void search(int key, NODE first)
{
NODE cur;
if(first==NULL)
{
printf("List is empty\n");
return;
}
cur = first;
while(cur != NULL)
{
if(key == cur->info)
break;
cur = cur->link;
}
if(cur==NULL)
{
printf("Search is unsuccesful \n");
return;
}
printf("Search is successful. Element is found in the list at %d",
                                                    cur->link);
}
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 main()
{
int item, choice, count, key, option;
NODE first = NULL;
for(;;)
{
printf("\n1. Insert front \n2. Delete rear \n3. Display list \n
4. Count items \n5. Search item \n6. Order list \n7. Exit \n");
printf("Enter your choice \n");
scanf("%d", &choice);
switch(choice) {
```

```
case1: printf("Enter the item at front end \n");
       scanf("%d", &item);
       first = insertfront(first, item);
       break;
case2: first = deletirear(first);
       Break;
case3: display(first);
       break;
case4: count = length(first);
       printf("Length of items in the list is %d \n", count);
       break;
case5: printf("Enter the item to be searched \n");
       scanf("%d", &key);
       search(key, first);
       break;
case6: printf("1. Ascending order \n 2. Descending order \n");
       scanf("%d", &option);
       if (option == 1)
         {
           first=asc(first);
           display(first);
         }
       else {
         first = desc(first);
         display(first);
         }
       break;
       default: exit(0);
       }}
  return 0;
}
```