```
#include<stdio.h>
#include<conio.h>
#include<alloc.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 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_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;
}
```

```
int   count (NODE first)
{
 NODE curr = first;
  int c ;
  if (first == NULL){
      c = 0;
      return c;
   }
   if (first->link == NULL){
      c = 1;
     return c;
   }
      while(curr != NULL){
        c++;
   }  curr = curr->link ;
      return c; }
```

```
void search( int key, NODE first)
{ int flag=0;
NODE cur;
if (first == NULL)
  {
     printf("list is empty\n");
     return;
  }
cur = first;
while(cur != NULL)
{
```

```
if (key == cur→info) { flag = 1;
  break;
cur = cur→link;
}
if ~~(cur == NULL)~~ (flag == 0)
  {
  printf ("search is unsuccessful \n");
   return;
  }
  printf ("search successful \n");
}
```

```
NODE  ASC (NODE first)
{
 NODE prev = first;
 NODE cur = NULL;
 int temp;
 if (first == NULL) {
     returno;
   }
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;
}
```

```
        DES
NODE  (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;
  }
```

```
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);
 }
}
```

papergrid
Date / /

```
void main()
{
int item, choice, choice2, j, key;
NODE first = NULL;
for(;;) {
printf("\n1: insertfront \n 2: deleterear
  \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 & 2 for
        descending order \n");
scanf("%d", &choice2);
 if ( choice2 ==1)
   first = asc (first);
if ( choice2 ==2)
  first = des (first);
```

Date: / /

```
break;
case 5:
printf ("enter the element to be
          searched for \n");
scanf ("%d", & key);
break;
case 6:
   display (first);
 break;
   default: exit (0);
   break;
}
}
getch ();
}
```