```
break;
case 6:
 first=order_list(first)
break;
default : exit (0);
}
}
}
```

```
int main (){
int item, choice, pos, i, n;
 NODE first =NULL;
 for (;;)
 {
printf ("1. insert-front \n2. delete_rear \n
   3. display \n 4. count_items \n5. search
\n 6. order \n Any other key to exit \n");
printf ("enter the choice \n");
scanf ("%d", &choice \n");
 scanf("%d
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_rear(first;
 break;
 case 4: count (first);
 break
case 5: printf ("Enter element to be
searched : ");
 scanf ("%d", &item);
```

```
int c=0, f=0;
while (temp1 != NULL) {
    c++;
    if (temp -> info == key) {
     printf ("search unsuccessful, eleme
     post" : %d \n", c);
        f=1; break;
    }
    temp = temp -> link;
}
 if (f==0)
    printf ("Search Unsuccessful !\n");
}
void display (NODE first)
 {
  NODE temp;
  if (first == NULL)
  printf ("list empty cannot display
 items \n");
 for (temp = first; temp1 != NULL; temp =
    temp -> link)
  {
 printf ("%d \n", temp -> info);
 }
}
```

```
int temp = ptr1 → info;
ptr1 → info = ptr1 → link → info;
ptr1 → link → info = temp
  swapped = 1;
}
 ptr1 = ptr1 → link;
}}
  lptr = ptr1;
 }}
  while (swapped);
    return first;
}
void count (NODE first)
{
  NODE temp;
   temp = first;
   int c = 0;
   while (temp != NULL)
   {
   temp = temp → link;
   c++;
  }
void list_search (NODE first, int key){
  NODE temp;
   temp = 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 (NODE first)
 {
   int swapped, i;
    NODE ptr1, lptr = NULL;
  if (first == NULL
     return first;
do
{
  swapped = 0;
  ptr = first;
  while (ptr1 -> link != lptr
  {
  if (ptr1 -> info > ptr1 -> link -> info)
  {
```

```
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;
```

linked list

```
#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;
}
NODE insert_front (NODE first, int item)
{
NODE temp;
temp = getnode();
```