LAB 8

```
#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 item;
    temp = getnode();
    temp -> info = item;
    temp -> link = NULL;
    if (first == NULL)
        return temp;
    temp -> link = first;
    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;
}
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);
    }
}
void count (NODE first)
{
    NODE temp;
```

```
int n=0;
if (first == NULL)
    printf ("list empty cannot display items \n");
else
{
    for (temp = first ; temp != NULL ; temp = temp -> link)
    {
        n++;
    }
    printf ("count = %d", n);
}
}
void search (int item , NODE first)
{
NODE temp;
int n=0 , t , c;
int flag = 1;
for ( temp = first; temp != NULL ; temp = temp -> link)
{
  n++;
}
c=n;
if (first == NULL)
    printf ("list empty cannot display items \n");
else
{
    for(temp = first ; temp != NULL ; temp = temp -> link)
```

```
{   {
        c--;
        t= temp -> link;
        if (item == t)
        {
            printf("\n item %d found at pos %d",
                    item, c);
            break;
            flag = 0;
        }
    }
    if (flag == 0) {
        printf("\n item does not exist\n");
    }
  }
}
NODE order_list (NODE first)
{
    int swapped, i;
    NODE ptr1, lptr = NULL;
    if (first == NULL)
    return first;
    do
    {
        swapped = 0;
        ptr1 = first;
```

```
        while (ptr1->link != lptr)
        {
            if (ptr1->info > ptr1->link->info)
            {
                temp = ptr1->info;
                ptr1->info = ptr1->link->info;
                ptr1->link->info = temp;
                swapped = 1;
            }
            ptr1 = ptr1->link;
        }
        lptr = ptr1;
    }
    while (swapped);
    return first;
}

int main()
{
    int item, choice, pos;
    NODE first = NULL;
    for (;;)
    {
        printf("\n1. Insert Front \n2. Insert Rear \n3. Delete Rear \n4. Count \n5. Search \n6. Order \n7. Exit
        \n\n");
```

```
printf("enter the choice \n");
scanf("%d", &choice);
switch(choice)
{
    case1: printf("enter the item at front \n");
           scanf("%d", &item);
           first = insert_front(first, item);
           break;
    case 2: first = delete_rear(first);
            break;
    case 3: display(first);
            break;
    case 4: count(first);
            break;
    case 5: printf("Enter item to be searched \n");
            scanf("%d", &item);
            search(item, first);
            break;
    case6: first = order_list(first);
           break;
    default: exit(0)
           break;
  }
}
```