Lab - Program - 9

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
   int info;
   struct node *rlink;
   struct node *llink;
};
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 dinsert_front(int item, NODE head)
{
  NODE temp, cur;
  temp = getnode();
  temp->info = item;
  temp->llink = NULL;
  temp->rlink = NULL;
  cur = head->rlink;
  head->rlink = temp;
  temp->llink = head;
  temp->rlink = cur;
  cur->llink = temp;
  return head;
}
NODE dinsert_rear(int item, NODE head)
{
  NODE temp, cur;
  temp = getnode();
  temp->info = item;
  temp->llink = NULL;
  temp->rlink = NULL;
  cur = head->llink;
  head->llink = temp;
  temp->rlink = head;
  cur->rlink = temp;
  temp->llink = cur;
  return head;
}
```

```
NODE ddelete_front(NODE head)
{
  NODE cur, next;
  if(head->rlink == head)
  {
    Printf("List is empty\n");
    return head;
  }
  cur = head->rlink;
  next = cur->rlink;
  head->rlink = next;
  next->llink = head;
  Printf("Item deleted at the front end
          is : %d\n", cur->info);
  free(cur);
  return head;
}

NODE ddelete_rear(NODE head)
{
  NODE cur, Prev;
  if(head->rlink == head)
  {
    Printf("List is empty\n");
    return head;
  }
  cur = head->llink;
  Prev = cur->llink;
  Prev->rlink = head;
  head->llink = Prev;
```

```
Printf("Item deleted at the rear end is : %d\n",
                                    cur->info);
  free(cur);
  return head;
}
void ddisplay(NODE head)
{
  NODE temp;
  if (head->rlink == head)
  {
    Printf("List is empty\n");
  }
  Printf("The contents of the list are:\n");
  temp = head->rlink;
  while(temp != head)
  {
    Printf("%d\n",temp->info);
    temp=temp->rlink;
  }
}
void dsearch(int key, NODE head)
{
  NODE cur;
  int count;
  if (head->rlink==head)
  {
    Printf("List is empty\n");
  }
```

```
cur=head->rlink;
count=1;
while(cur!=head && cur->info!=key)
{
  cur=cur->rlink;
  count++;
}
if(cur==head)
{
  Printf("search unsuccessfull\n");
}
else
{
  Printf("key element found at the position %d\n",
}
}

NODE dinsert_leftPos(int item, NODE head)
{
  NODE cur, prev, temp;
  if(head->rlink==head)
  {
    Printf("List is empty\n");
    return head;
  }
  cur=head->rlink;
  while(cur!=head)
  {
    if(cur->info==item)
    {
      break;
    }
```

```
        cur=cur->rlink;
}
 if(cur==head)
 {
   Printf("No such item found in the list\n");
    return head;
 }
   Prev=cur->llink;
   temp=getnode();
   temp->llink=NULL;
   temp->rlink=NULL;
     Printf("Enter the item to be inserted at the
                left of the given item\n");
      Scanf("%d",&temp->info);
       Prev->rlink=temp;
      temp->llink=Prev;
      temp->rlink=cur;
       cur->llink=temp;
        return head;
 }
 NODE dinsert_rightpos(int item, NODE head)
 {
    NODE temp, cur, next;
```

```
if(head->rlink==head)
{
 Printf("List is empty\n");
  return head;
 }
 cur=head->rlink;
 while(cur!=head)
 {
  if(cur->info==item)
  {
   break;
  }
  cur=cur->rlink;
}
if(cur==head)
{
 Printf("No such item found in the list\n");
 scanf("%d",&temp->info);
  return head;
}
next=cur->rlink;
temp=getnode();
temp->llink=NULL;
temp->rlink=NULL;
Printf("Enter the item to be inserted at
        the right of the given item:\n");
```

```
    scanf("%d", &temp->info);
    cur->rlink=temp;
    temp->llink=cur;
    next->llink=temp;
    temp->rlink=next;
    return head;
}
NODE ddelete_duplicates(int item, NODE head)
{
    NODE Prev, cur, next;
    int count=0;
    if(head->rlink==head)
    {
        Printf("List is empty\n");
        return head;
    }
    cur=head->rlink;
    while(cur!=head)
    {
        if(cur->info!=item)
        {
            cur=cur->rlink;
        }
        else
        {
            count++;
            if(count==1)
            {
                cur=cur->rlink;
            } continue;
```

```
else
{
 Prev = cur -> llink;
 next = cur -> rlink;
  Prev -> rlink = next;
  next -> llink = Prev;
  free(cur);
   cur = next;
 }

}
}
 if (count == 0)
 {
  Printf("No such item found in the list\n");
 }
 else
 {
  Printf("All the duplicate elements of the
      given item are removed successfully\n");
   }
   return head;
}

 NODE delete_all_key(int item, NODE head)
 {
  NODE Prev, cur, next;
  int count;
```

```
if (head -> rlink == head)
{
  Printf("LE");
  return head;
}
count = 0;
cur = head -> rlink;
while (cur != head)
{
  if (item != cur -> info)
  cur = cur -> rlink;
  else
  {
    count++;
    Prev = cur -> llink;
    next = cur -> rlink;
    Prev -> rlink = next;
    next -> llink = Prev;
    freenode (cur);
    cur = next;
  }
}
if (count == 0)
Printf ("key not found");
else
Printf ("key found at %d Positions and are deleted\n", count);
return head;
}
```

```
int main()
{
NODE head;
int item, choice, key;
head=getnode();
head->llink=head;
head->rlink=head;
for(;;)
{
Printf("\n1:dinsert front\n2:dinsert rear\n3:
d delete front\n4:ddelete rear\n5:ddisplay\n6:
dsearch\n7:dinsert leftPos\n8:dinsert rightPos\n9:
ddelete duplicates\n10:ddelete_based on
specified value\n11:exit\n");
Printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: Printf("Enter the item at front
        scanf("%d",&item);
        head=dinsert_front(item,head);
        break;
case 2: Printf("Enter the item at rear end
        scanf("%d",&item);
        head=dinsert_rear(item,head);
```

```
case 3 : head = ddelete_front(head);
         break;

case 4 : head = ddelete_rear(head);
         break;

case 5 : ddisplay(head);
         break;

case 6 : Printf("Enter the key element to be
                 Searched: \n");
         scanf("%d", &key);
         dsearch(key, head);
         break;

case 7 : Printf("Enter the key element: \n");
         scanf("%d", &key);
         head = dinsert_leftPos(key, head);
         break;

case 8 : Printf("Enter the key element: \n");
         scanf("%d", &key);
         head = dinsert_rightPos(key, head);
         break;

case 9 : Printf("Enter the key element whose
         duplicates should be removed: \n");
         scanf("%d", &key);
         head = ddelete_duplicates(key,
         break;                    head);
```

```
case 10 : Printf("Enter the key value\n");
          scanf("%d", &item);
          delete_all_key (item, head);
          break;
case 11 : exit(0);
default : Printf("Invalid choice\n");
}
}
return 0;
}
```