Lab-program

Doubly linked list 14/12/202

```
#include<stdio.h>
# include<stdlib.h>
 struct node
 {
 int info;
  struct node*llink;
  struct node*rlink;
   };
typedef struct node*NODE;
 NODE getnode()
 {
 NODE x;
 x=(NODE)malloc(sizeof(structnode));
   if(x==NULL)
 {
 printf("memory full");
 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;
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;
cur = head->llink
head->llink = temp.
temp->rlink = head;
temp->llink = cur;
cur->rlink = temp;
return head;
}
NODE ddelete_rear ( NODE head)
{
NODE cur, prev;
if (head->rlink = = head)
{
printf (" empty\n");
return head;
}
```

```
cur = head->link;
prev = cur->link;
head->link = prev
prev->rlink = head;
printf("The node deleted is %d", cur->info);
  freenode(cur);
return head;
}
NODE insert_leftpos(int item, NODE head)
{ NODE temp, cur, prev;
if (head->rlink = = head)
{
printf("list empty\n");
 return head;
}
cur = head->rlink;
while (cur! = head)
{
if (item = = cur->info) break;
  cur = cur->rlink;
}
if (cur= = head)
{
printf(" key not found\n");
 return head;
}
prev = cur->llink;
printf("enter towards left of %d = ", item);
```

```
temp getnode();
scanf(" %d", &temp->info);
prev->rlink = temp;
temp->llink = prev;
cur->link = temp;
temp->rlink = cur;
return head;
}
NODE insert_rightpos (int item, NODE head)
{
NODE temp, cur, next;
if(head->rlink = =head)
{
printf (" list empty\n");
   return head;
}
cur =head->rlink;
while (cur!=head)
{
if (item= cur ->info) break;
   cur = cur->rlink;
}
if (cur==head)
{
printf (" Key not found\n");
 return head;
}
next= cur->rlink;
```

```
printf ("enter towards right of %d =", item);
temp= getnode;
scanf= ("%d", &temp->info);
cur->rlink=temp;
temp->llink= cur;
next->llink= temp;
temp->rlink =next;
return head;
}
NODE search(NODE head, int item)
{
NODE temp, cur;
int flag=0;
if (head->rlink==head)
{
printf (" list empty\n");
return head;
}
cur= head->rlink;
while (cur!= head)
{
if (item==cur->info)
{
flag=1;
Break;
}
cur= cur->rlink;
}
```

```
if (cur= =head)
  printf ("search unsuccessfull|n");
   if (flag= =1)
  printf ("search successfull|n");
}
NODE delete_all_Key (int item, NODE head)
 {
 NODE prev, cur, next;
 int count ;
  if (head->rlink= =head)
 {
printf(" list empty|n");
 return head;
 }
 count =0;
 cur=head->rlink ;
 while (cur != head)
 {
 if (item != cur->info)
  cur=cur->rlink ;
   else
 {
 count ++;
  prev = cur-> link ;
  next= cur->rlink;
  prev->rlink =next;
  next->llink = prev;
   freenode(cur);
   cur= next ;
 }
```

```
  if (count = = 0)
   printf ("not found\n");
    else {
 printf ("found at %d positn & deleted", count);
     return head;
   }
 }
Void display(NODE head)
 {
 NODE temp;
  if (head-> rlink ==head)
  {
  printf ("dq is empty\n");
   return;
  }
  printf ("contents of dq\n");
  temp= head ->rlink;
  while (temp!= head)
 {
 printf (" %d", temp->info);
 temp= temp->rlink;
 }
 printf ("\n");
 }
 void main()
  {
  NODE head, last;
  int item, choice ;
```

```
head = getnode( )
head->rlink = head;
head-llink = head;
  for(;;)
{
printf("\n1.insert front\n2:insert rear\n 3:delete front\n4:
  delete rear\n5: insert left of key element\n6:
  insert right of key element\n 7:search\n 8: delete
  repeating elements\n 9: display\n 10: exit\n");
printf("enter the choice\n");
scanf("%d", &choice);
 switch (choice)
{
case 1 : printf(" enter the item at front end\n");
         scanf (" %d", &item);
         last = dinsert_front (item, head); break;
case 2 : printf ("enter the item at rear end\n");
         scanf ("%d", &item)
         last = dinsert_rear(item, head);
         break;
case 3; last = ddelete_front (head); break;
case 4 ; last = ddelete_rear (head) ;   break;
case 5 : printf (" enter the key element\n");
         scanf ("%d", & item);
         last = insert_leftpos(item, head);   break
case 6 : printf ("enter the key element\n");
         scanf("%d", &item);
         last = insert_rightpos (item, head);
case 7: printf ("enter the search element\n");
         scanf ("%d", &item);
        Search(head, item);
         break;
case 8: printf ("enter the element to be deleted for duplicate\n");
         scanf ("%d",&item);
        last = delete_all_key (item, head);
case 9: display (head);
         break;
 default : exit (0);
   }
 }
}
```

Scanned with CamScanner