PAGE No.
DATE 18/12/2020

WEEK-11 Lab Program 9

Q) WAP to implement doubly linked list with primitive operations.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *prev;
struct node *next;
};
typedef struct node *NODE;
int count;
NODE getnode()
{
NODE p;
p=(NODE)malloc(sizeof(struct node));
if(p==NULL)
{
printf("mem is full\n");
exit(0);
}
return p;
}
NODE insert_front(int item, NODE head)
{
NODE newn, cur;
newn=getnode();
newn->data=item;
newn->prev=NULL;
newn->next=NULL;
```

```
cur = head -> next;
 head -> next = newn;
 newn -> prev = head;
 newn -> next = cur;
  cur -> prev = newn;
return head;
}
NODE insert_rear(int item, NODE head)
{
 NODE newn, cur;
 newn = getnode();
 newn -> data = item;
 newn -> prev = NULL;
 newn -> next = NULL;
 cur = head -> prev;
 head -> prev = newn;
 newn -> next = head;
 cur -> next = newn;
 newn -> prev = cur;
 return head;
}
NODE insert_left(int item, NODE head)
{
   NODE newn, cur, left;
   if(head -> next == head)
      {
       printf("DLL is empty");
       return head;
       }
      else
      { cur = head -> next;
          while(cur != head)
```

```
{
if(cur->data==item)
 {
  break;
 }
 cur=cur->next;
}
if(cur==head)
 {
  printf("key node for insertion not found");
  return head;
}
left=cur->prev;
printf("Enter data towards left of %d",item);
new=getnode();
scanf("%d",&new->data);
new->prev=left;
left->next=new;
new->next=cur;
cur->prev=new;
return head;
}
}
NODE insert_right(int item, NODE head)
{
 NODE new, cur, right;
 if(head->next==head)
  {
   printf("DLL is empty");
  return head;
  }
 cur=head->next;
```

```
while (cur!=head)
{
 if(cur->data==item)
    {
     break;
     }
    cur=cur->next;
 }
if(cur==head)
 {
 printf("key node for insertion not found");
  return head;
}
right=cur->next;
 rem=getnode();
 printf("Enter data towards right of %d",item);
 scanf("%d",&rem->data);
 right->prev=rem;
 rem->next=right;
 rem->prev=cur;
 cur->next=rem;
 return head;
}
NODE delfront(NODE head)
 {
 NODE cur, next;
 if(head->next==head)
  {
   printf("DLL is empty");
   return head;
  }
```

```
cur=head->next;
next=cur->next;
head->next=next;
next->prev=head;
printf("The node deleted is :%d",cur->data);
free(cur);
return head;
}
NODE del_rear(NODE head)
{
NODE cur,left;
if(head->next==head)
{
printf("DLL is empty");
}
cur=head->prev;
left=cur->prev;
left->next=head;
head->prev=left;
printf("The node is deleted %d",cur->data);
free(cur);
return head;
}
NODE dll_search(int item,NODE head)
{
count=0;
NODE cur;
cur=head->next;
while(cur!=head)
{
count++;
if(cur->data==item)
```

```
{
printf("Item found at %d", count);
return head;
}
cur=cur->next;
}
if (count==0)
  {
  printf("Element not found");
  }
return head;
}
NODE del_all(int item, NODE head)
{
NODE left, cur, right;
if(head->next==head)
  {
  printf("DLL empty");
  return head;
  }
count=0;
cur=head->next;
while(cur!=head)
{
if(item!=cur->data)
  {
  cur=cur->next;
  }
else
{
count++;
left=cur->prev;
```

```
right = cur -> next;
left -> next = right;
right -> prev = left;
free(cur);
cur = right;
}
}
if (count == 0)
{
printf("key not found");
}
else
{
printf("key found at %d position and are
deleted", count);
}
return head;
}
NODE remove_duplicates(int item, NODE head)
{
NODE cur, left, right;
count = 0;
if (head -> next == head)
{
printf("DLL is empty");
}
cur = head -> next;
while (cur != head)
{
if (cur -> data == item)
{
count++;
```

```
if(count>1)
  {
  left=cur->prev;
  right=cur->next;
  left->next=right;
  right->prev=left;
  free(cur);
  cur=right;
  }
  else
  {
  cur=cur->next;
  }
}
else if(cur->data!=item)
  {
  cur=cur->next;
  }
}
if(count==1)
{
printf("given key has no duplicates");
}
else
 {
 printf("Duplicates have been removed
only unique present");
 }
return head;
}
void display(NODE head)
{
```

```
NODE cur;
  if(head->next==head)
  {
  printf("Doubly linked list is empty");
  return;
  }
cur=head->next;
while(cur!=head)
  {
  printf("%d", cur->data);
  cur=cur->next;
  }
printf("\n");
}
int main()
  {
  NODE head;
  head=getnode();
  head->next=head;
  head->prev=head;
NODE p;
  int ch, item;
  int i, c, val;
for(;;)
  {
printf("\n1. Insert front \n2. Insert rear \n3.
...Insert left of the given node \n4. insert
right of given node \n5. delete front \n6.
delete rear \n7. delete all occurrence of key
8. display \n9. searching for a given element
\n10. remove duplicates");
printf("\nPlease enter your choice");
```

```
scanf("%d", &ch);
switch(ch)
{
case 1:
    printf("Enter the item:");
    scanf("%d", &item);
    head = insert_front(item, head);
    break;

case 2: printf("Enter the key item");
    scanf("%d", &item);
    head = insert_rear(item, head);
    break;
case 3: printf("Enter the key item\n");
    scanf("%d", &item);
    head = insert_left(item, head);
    break;
case 4: c=0;
    printf("Enter the key item\n");
    scanf("%d", &item);
    head = insert_right(item, head);
    break;
case 5: del_front(head);
    break;
case 6: del_rear(head);
    break;
case 7: printf("Enter the key to be deleted");
    scanf("%d", &item);
    head = del_all(item, head);
    break;
case 8: display(head);
    break;
```

```
case 9: printf("Enter the value to be searched");
        scanf("%d", &item);
        head = dll_search(item, head);
        break;
case 10: printf("Enter the key whose duplicate need
        to be deleted");
        scanf("%d", &item);
        head = remove_duplicates(item, head);
        break;
default: exit(0);
}
}
return 0;
}
```