→ WAP to implement doubly linked list with following operations: a) Create a doubly linked list.
b) Insert front & rear.
c) Delete the node both front & rear.
d) Display the contents of the list.
e) Simple search
f) Inserting the node before and after key node.
g) Deleting repeating occurances.

```
#include<stdio.h>
#include<conio.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(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;
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_front(NODE head)
{
NODE cur, next;
if(head->rlink == head)
{
printf("linked list is empty\n");
return head;
}
cur = head->rlink;
next = cur->rlink;
head->rlink = next;
next->llink = head;
printf("The node deleted is %d", cur->info);
freenode(cur);
return head;
}
NODE ddelete_rear(NODE head)
{
NODE cur, prev;
```

```
if (head -> rlink == head)
{
printf("linked list is empty\n");
return head;
}
cur = head -> llink;
prev = cur -> llink;
head -> llink = 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 is 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 -> llink = temp;
temp -> rlink = cur;
return head;
}
NODE insert_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 (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 ("linked 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 unsuccessful\n");
if (flag == 1)
printf("search successful\n");
}
NODE delete_all_key(int item, NODE head)
{
NODE prev, cur, next;
int count;
if (head -> rlink == head)
{
printf("List is empty\n");
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("Element is not found\n");
else { printf("found at %d positions and are deleted",
```

```
return head; }}
void display (NODE head)
{
NODE temp;
if (head -> rlink == head)
{
printf (" Linked list is empty \n");
return; }
printf (" Contents of doubly linked list : \n");
temp = head -> rlink;
while (temp != head)
{
printf ("%d ", temp -> info );
temp = temp -> rlink;
}}
void main()
{
NODE head, last;
int item, choice;
head = getnode ();
head -> rlink = head;
head -> llink = head;
for (;;)
{
printf ("\n1:Insert front \n 2. Insert Rear \n 3. Delete front \n 4. Delete
rear \n 5. Insert to the left of the key element \n 6. Insert to the
right of the key element \n 7. Search \n 8. Delete duplicate items
\n 9. Display \n 10. Exit \n Enter your choice:");
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);
        break;
case 7: printf("enter the search element:\n");
        scanf("%d", &item);
        search(head,item);
        break;
case 8: printf("Enter the element to be deleted\n");
        scanf("%d", &item);
        last= delete_all_key(item, head);
        break;
case 9: display(head);
        break;
default: exit(0);
}}}
```