

LAB PROGRAM-9

9) WAP Implement doubly link list with primitive operations a) Create a doubly linked list. b) Insert a new node to the left of the node. c) Delete the node based on a specific value d) Display the contents of the list.

CODE:

```
#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 unsuccessful\n");
}
else
{
printf("Key element found at the position %d\n",count);
}
}

```

```

}
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 dinser_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");
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:ddelete
front\n4:ddelete rear\n5:ddisplay\n6:dsearch\n7:dinsert
lestpos\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 end:\n");
scanf("%d",&item);
head=dinsert_front(item,head);
break;
case 2: printf("Enter the item at rear end:\n");
scanf("%d",&item);
head=dinsert_rear(item,head);
break;
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,head);
break;
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;
}
```

OUTPUT :

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
1

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
1
Enter the item at front end:
2
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

1

Enter the item at front end:

3

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

5

The contents of the list are:

3

2

1

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
7
Enter the key element:
2
Enter the item to be inserted at the left of the given item:
4

1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
4
2
1
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
8
Enter the key element:
4
Enter the item to be inserted at the right of the given item:
5
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
4
5
2
1
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
2
Key element found at the position 4
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
6
Enter the key element to be searched:
9
Search unsuccessfull
```



```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

2

Enter the item at rear end:

2

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

5

The contents of the list are:

3

4

5

2

1

2

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
9
Enter the key element whose duplicates should be removed:
2
All the duplicate elements of the given item are removed successfully
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
4
5
2
1
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
10
Enter the key value
5
Key found at 1 positions and are deleted
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
5
The contents of the list are:
3
4
2
1
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
3
Item deleted at the front end is:3
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
4
Item deleted at the rear end is:1
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
3
Item deleted at the front end is:4
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
4
Item deleted at the rear end is:2
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
Enter the choice
3
List is empty
```

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

30

Invalid choice

```
1:dinsert front
2:dinsert rear
3:ddelete front
4:ddelete rear
5:ddisplay
6:dsearch
7:dinsert lestpos
8:dinsert rightpos
9:ddelete duplicates
10:ddelete_based on specified value
11:exit
```

Enter the choice

11

Process returned 0 (0x0) execution time : 616.916 s

Press any key to continue.

