```
//WAP Implement doubly link list with following operations: Create a doubly linked list.
//b)	Insert front and rear.
//c)	Delete the node both front and rear	d) Display the contents of the list
//e) Simple search f)Inserting the node before and after the key node g)Deleting all Occurrences.
#include<stdio.h>
#include<conio.h>
#include<process.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("mem 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("dq 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("dq 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;
}
void display(NODE head)
```

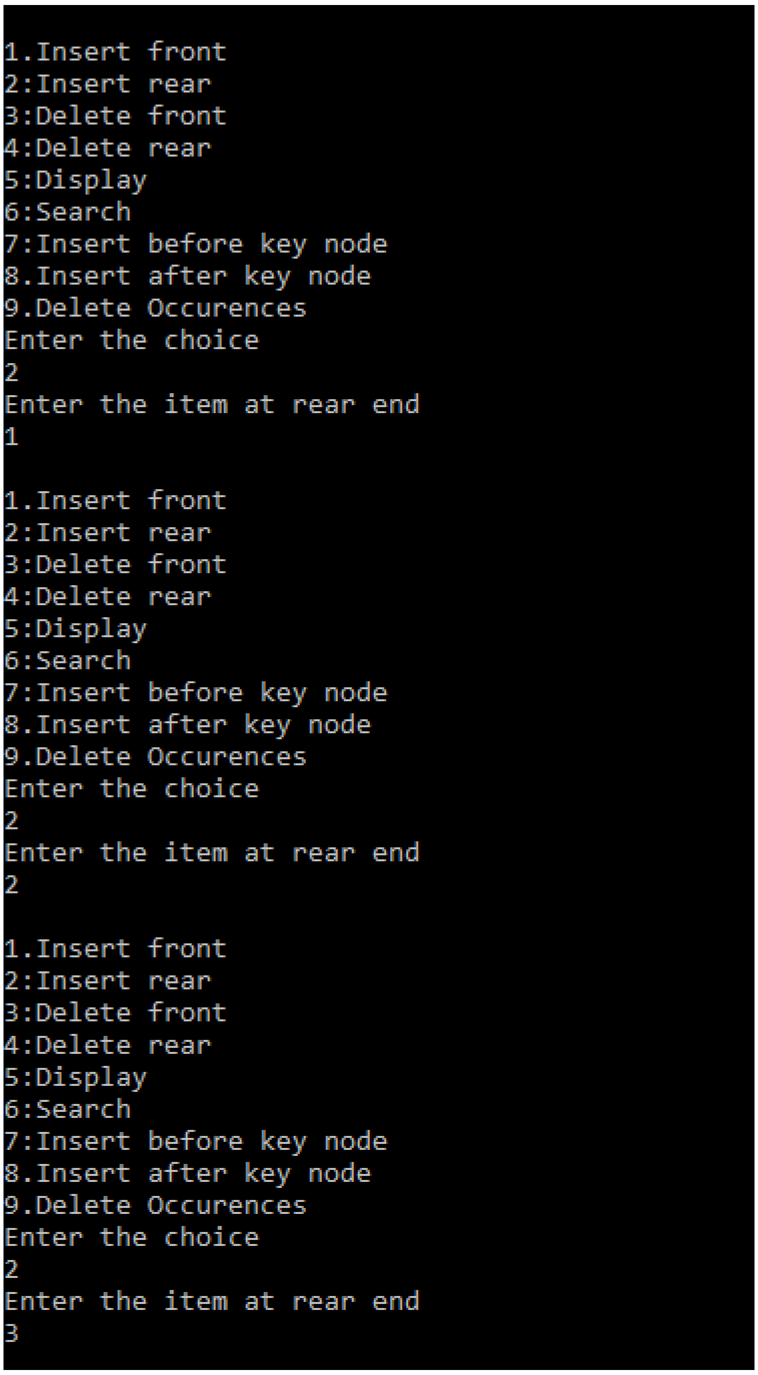
```
{
NODE temp;
if(head->rlink==head)
{
printf("dq empty\n");
return;
}
printf("contents of List: \n");
temp=head->rlink;
while(temp!=head)
{
printf("%d  ",temp->info);
temp=temp->rlink;
}
printf("\n");
}
void search(NODE head)
{
        bool flag=false;
        NODE x=head->rlink->llink;//head's address
        NODE temp=head->rlink;
        int item,count=1;
        printf("Enter the element to be searched in the list: ");
        scanf("%d",&item);
        while(temp!=x)
        {
                if(temp->info==item)
                {
                        flag=true;
                        break;
                }
                temp=temp->rlink;
                count++;
        }
        if(flag)
        printf("Element is found in the list at position %d",count);
        else if(temp==x)
        printf("Element not found\n");
}
NODE insert_leftpos(NODE head)
{
NODE temp,cur,prev;
int item;
printf("Enter the item");
scanf("%d",&item);
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->llink=temp;
 temp->rlink=cur;
 return head;
}

NODE insert_rightpos(NODE head)
{
		NODE temp,cur,next;
int item;
printf("Enter the item");
scanf("%d",&item);
if(head->rlink==head)
{
printf("list is empty\n");
return head;
}
cur=head->rlink;
while(cur!=head)
{
if(item==cur->info)break;
```

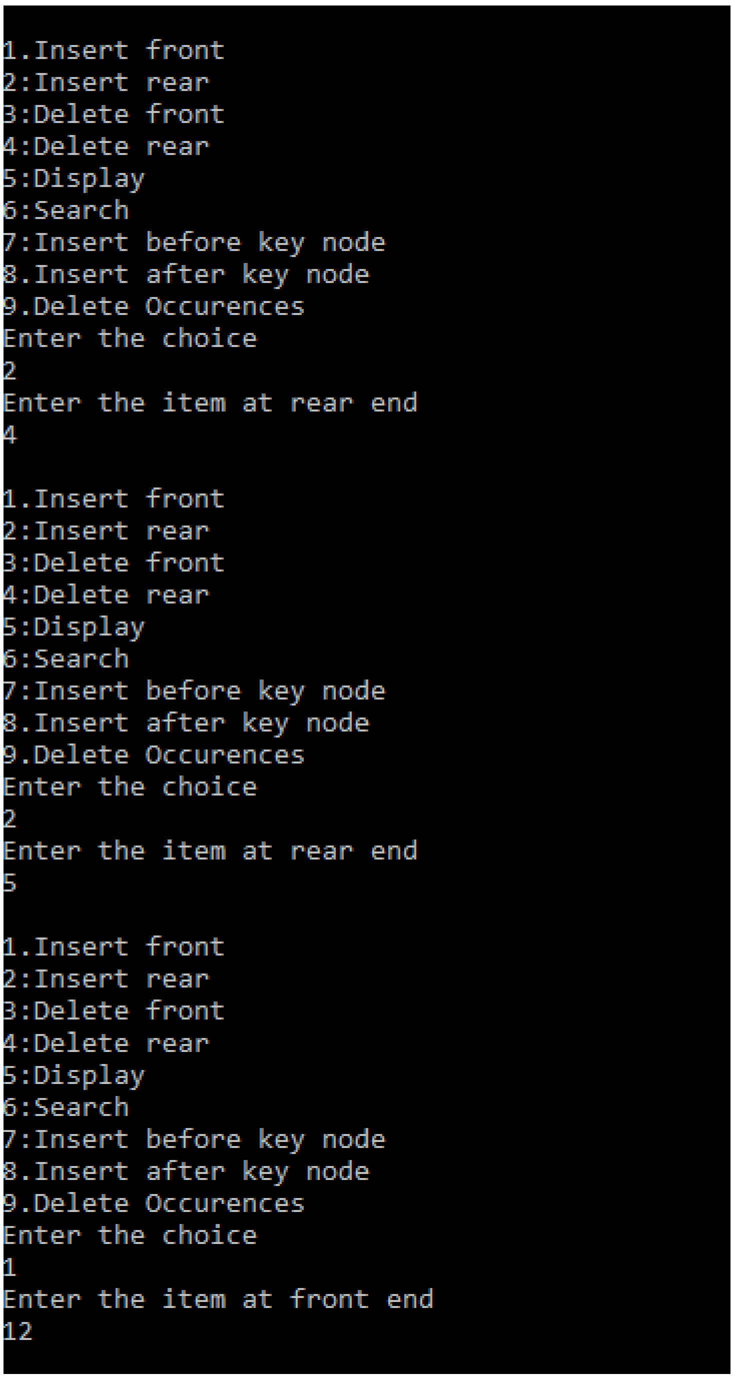
```
cur=cur->rlink;
}
if(cur==head)
{
 printf("Key not found\n");
 return head;
 }
 next=cur->rlink;
 printf("enter item towards right of %d=",item);
 temp=getnode();
 scanf("%d",&temp->info);
 next->llink=temp;
 temp->rlink=next;
 cur->rlink=temp;
 temp->llink=cur;
 return head;

}



NODE delete_all_key(NODE head)
{
NODE prev,cur,next;
int item;
printf("Enter the item: ");
scanf("%d",&item);
int count;
   if(head->rlink==head)
    {
     printf("Linked list 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("key not found\n");
  else
 printf("key found at %d positions and are deleted\n", count);


return head;
}
int main()
{
```

```
NODE head,last,y;

int item, choice;

head=getnode();

head->rlink=head;

head->llink=head;

for(;;)

{

        printf("\n1.Insert front\n2:Insert rear\n3:Delete front\n4:Delete rear\n5:Display\n6:Search\n7:Insert 
before key node\n8.Insert after key node\n9.Delete Occurences\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: display(head);

                        break;

                case 6: search(head);

                   break;
```

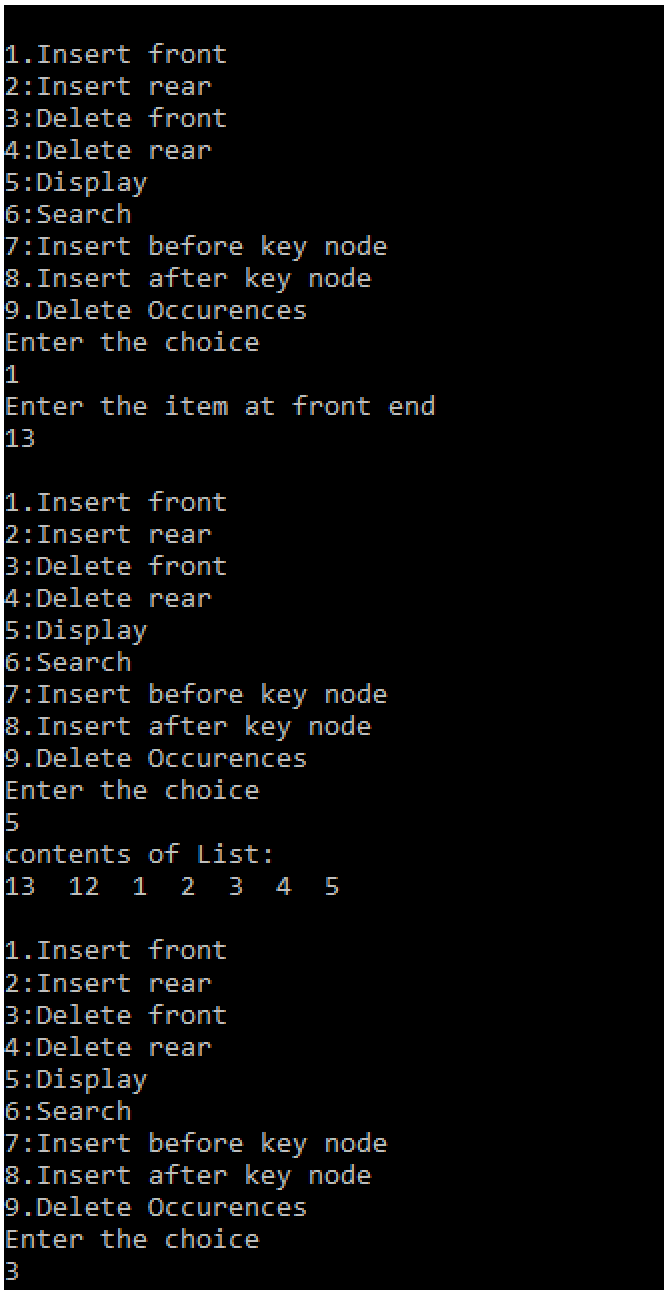
```
                    case 7:y=insert_leftpos(head);
                        break;
                    case 8: y=insert_rightpos(head);
                          break;
                    case 9:delete_all_key(head);
                                break;
                    default:exit(0);
                    }
          }
}
```

OUTPUT:

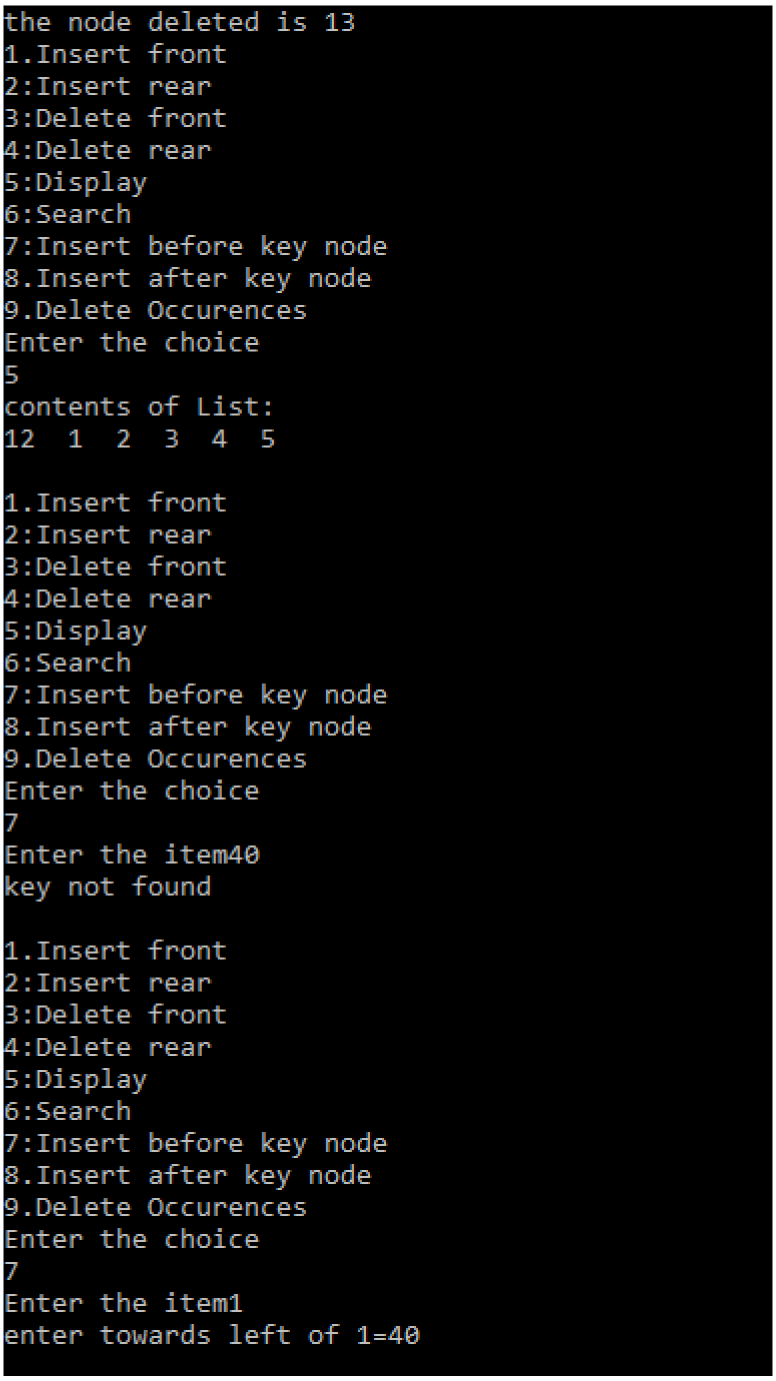

```

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
1

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
2

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
3
```



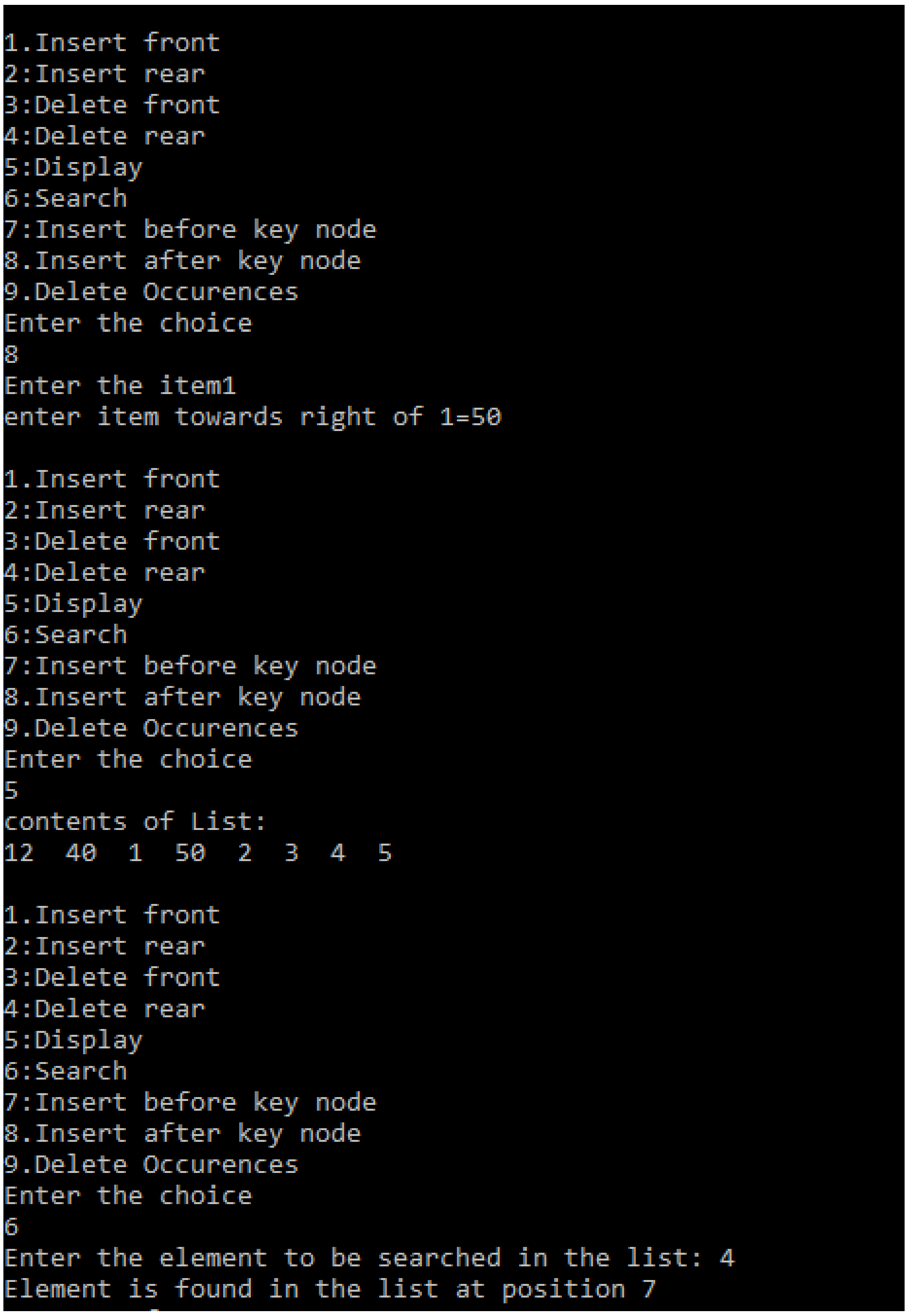
```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
4

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
2
Enter the item at rear end
5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
1
Enter the item at front end
12
```



```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
1
Enter the item at front end
13

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
13  12  1  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
3
```



```
the node deleted is 13
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
12  1  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
7
Enter the item40
key not found

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
7
Enter the item1
enter towards left of 1=40
```



```
1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
8
Enter the item1
enter item towards right of 1=50

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
5
contents of List:
12  40  1  50  2  3  4  5

1.Insert front
2:Insert rear
3:Delete front
4:Delete rear
5:Display
6:Search
7:Insert before key node
8.Insert after key node
9.Delete Occurences
Enter the choice
6
Enter the element to be searched in the list: 4
Element is found in the list at position 7
```