EXPERIMENT NO. 6

IMPLEMENTATION OF LINKED LIST

Aim: Write A Program to Implement Linked List.

Theory: A ***Linked List*** is a linear collection of data elements, called ***nodes***, each pointing to the next node by means of a Pointer. It is a Data Structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of data and a Reference **(in other words, a link)** to the next node in the sequence.

Linked lists are among the simplest and most common data structures. They can be used to implement several other common ADTs, including lists (the abstract data type), Stacks, Queue, Associative Arrays, and S-Expressions, though it is not uncommon to implement the other data structures directly without using a list as the basis of implementation.

The Principal Benefit of a Linked List over a Conventional Array is that the list elements can easily be inserted or removed without reallocation or reorganization of the entire structure because the data items need not be stored contiguously in memory.

Algorithm:

FOR APPEND OPERATION:

```
1. Start
2. Declare Structure Pointer Temp and R
3. if (Q= =NULL) (Step 4 To Step 9)
4. Display “Creating List”.
5. Create A Structure of Memory Requirement Node and Store the Starting
   Address in Temp
6. Place The Data (Num) in Temp’s Data Section
7. Place “NULL” In Temp’s Link Section
8. Copy Temp in P
9. Copy Temp in Q
10. Else (Step 10 To Step 17)
11. Copy Q in R
12. Repeat Step 13 while(r->link! =NULL)
```

```
13. R=R- > link
14. Create A Structure of Memory Requirement Node and Store the Starting
    Address in Temp
15. Place The Data (Num) in Temp's Data Section
16. Place "NULL" In Temp's Link Section
17. Copy Temp in R's Link Section
18. Stop
```

FOR INBEGIN OPERATION:

```
1.  Start
2.  Declare Structure Pointer Temp
3.  if (Q= =NULL)
4.  Display "List Does Not Exists".
5.  Else (Step 6 To Step 9)
6.  Create A Structure of Memory Requirement Node and Store the
    Starting Address in Temp
7.  Place The Data (Num) in Temp's Data Section
8.  Place P in Temp's Link Section
9.  Copy Temp in P
10. Stop
```

FOR DELETE OPERATION:

```
1.  Start
2.  Declare Structure Pointer Old
3.  Declare Found =0
4.  if (Q= =NULL)
5.  Display "List Does Not Exists".
6.  Else (Step 6 To Step 23)
7.  Repeat till Step 19 while (Q! =NULL)
8.  If (Q's Data Section = =Num) (Step 8 To Step 16)
9.  If (Q= =P) (Step 10 To Step 12)
10. P=Q's Link Section
11. Change Found to 1
12. Q=Q's Link Section
13. Else (Step 13 To Step 16)
14. Old's Link Section = Q's Link Section
15. Change Found to 1
```

```
16.Q=Q’s Link Section
17.Else (Step 17 To Step 19)
18.Old=Q
19.Q=Q’s Link Section
20.If (Found = =1)
21.Display “Number Deleted”
22.Else
23.Display “Number Not Found”
24.Stop
```

FOR COUNT OPERATION:

```
1.  Start
2.  Declare C =0
3.  if (Q= =NULL)
4.  Display “List Does Not Exists”
5.  Else (Step 5 To Step 10)
6.  Repeat till Step 8 while (Q! =NULL)
7.  Increment C by 1
8.  Q=Q’s Link Section
9.  Display Value of C
10.Stop
```

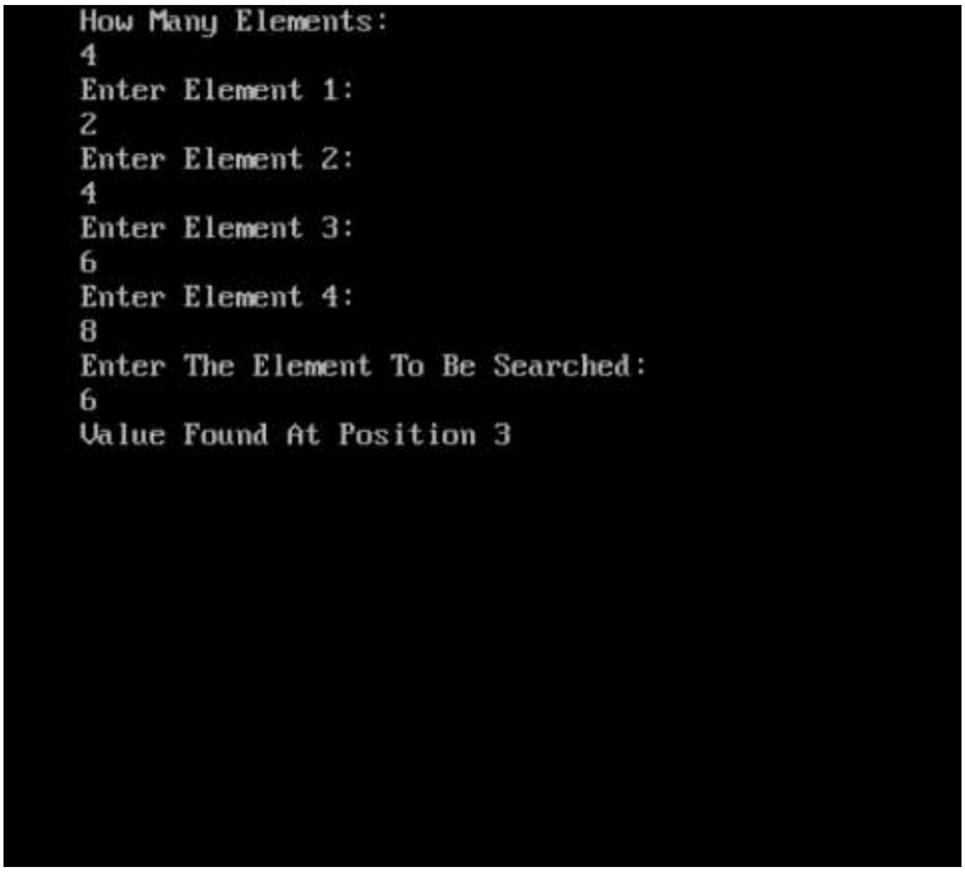
FOR DISPLAY OPERATION:

```
1.  Start
2.  if (Q= =NULL)
3.  Display “List Does Not Exists”
4.  Else (Step 4 To Step 8)
5.  Display “Contents”
6.  Move to A New Line
7.  Repeat till Step 8 while (Q! =NULL)
8.  Display Value of (Q’s Data Section)
9.  Move to A New Line
10.Q=Q’s Link Section
11.Stop
```

Program:

```
#include<stdio.h>
int main()
{
int
a[100],n,key,low,mid,
high,i,chk=0;
printf("How Many
Elements:\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter Element
%d:\n",(i+1));
scanf("%d",&a[i]);
}
printf("Enter The
Element To Be
Searched:\n");
scanf("%d",&key);
low=0; high=n-1;
while(low<=high)
{
mid=(low+high)/2;
if(a[mid]==key)
{
printf("Value Found
At Position
%d\n",(mid+1));
chk=1;
break;
}
if(a[mid]>key)
high=mid-1;
else low=mid+1;
}
if(chk==0)
printf("Value Not
Found\n");
return 0;
}
```

Output:



```
How Many Elements:
4
Enter Element 1:
2
Enter Element 2:
4
Enter Element 3:
6
Enter Element 4:
8
Enter The Element To Be Searched:
6
Value Found At Position 3
```