

**DSA LAB**  
**Experiment number 06**

**Name:** Aamir Ansari

**Batch:** A

**Roll no.** 01

**Aim:** Implementation of singly linked list

**Theory:**

A singly linked list is a type of linked list that is unidirectional, that is, it can be traversed in only one direction from head to the last node (tail).

Each element in a linked list is called a **node**. A single node contains data and a pointer to the next node which helps in maintaining the structure of the list.

**Algorithms:**

Insert at Beginning:

Step 1: IF AVAIL = NULL, then  
    Write OVERFLOW  
    Go to Step 7  
    [END OF IF]

Step 2: SET New\_Node = AVAIL //allocate space for new node.

Step 3: SET AVAIL = AVAIL->NEXT

Step 4: SET New\_Node->DATA = VAL //Set data part with given value

Step 5: SET New\_Node->Next = START //Next part initialized with the address of first node of list

Step 6: SET START = New\_Node // Make new node as a Start node of list

Step 7: EXIT

\*\*\*

### Insert at End:

Step 1: IF AVAIL = NULL, then  
    Write OVERFLOW  
    Go to Step 10  
    [END OF IF]

Step 2: SET New\_Node = AVAIL

Step 3: SET AVAIL = AVAIL->NEXT

Step 4: SET New\_Node->DATA = VAL

Step 5: SET New\_Node->Next = NULL

Step 6: SET PTR = START

Step 7: Repeat Step 8 while PTR->NEXT != NULL

Step 8: SET PTR = PTR ->NEXT

    [END OF LOOP]

Step 9: SET PTR->NEXT = New\_Node

Step 10: EXIT

\*\*\*

Insert after value NUM:

Step 1: IF AVAIL = NULL, then  
    Write OVERFLOW  
    Go to Step 12  
    [END OF IF]

Step 2: SET New\_Node = AVAIL

Step 3: SET AVAIL = AVAIL->NEXT

Step 4: SET New\_Node->DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat Steps 8 and 9 while PREPTR->DATA != NUM

Step 8: SET PREPTR = PTR

Step 9: SET PTR = PTR->NEXT

    [END OF LOOP]

Step 10: PREPTR->NEXT = New\_Node

Step 11: SET New\_Node->NEXT = PTR

Step 12: EXIT

\*\*\*

Insert node before the value num:

Step 1: IF AVAIL = NULL, then  
    Write OVERFLOW  
    Go to Step 12  
    [END OF IF]

Step 2: SET New\_Node = AVAIL

Step 3: SET AVAIL = AVAIL->NEXT

Step 4: SET New\_Node->DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat Steps 8 and 9 while PTR->DATA != NUM

Step 8: SET PREPTR = PTR

Step 9: SET PTR = PTR->NEXT

    [END OF LOOP]

Step 10: PREPTR->NEXT = New\_Node

Step 11: SET New\_Node->NEXT = PTR

Step 12: EXIT

\*\*\*

### Deleting first node:

Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 5  
    [END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START->NEXT

Step 4: FREE PTR

Step 5: EXIT

\*\*\*

### Deleting last node:

Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 8  
    [END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Steps 4 and 5 while PTR->NEXT != NULL

Step 4:  
SET PREPTR = PTR

Step 5:  
SET PTR = PTR->NEXT

[END OF LOOP]

Step 6: SET PREPTR->NEXT = NULL

Step 7: FREE PTR

Step 8: EXIT

\*\*\*

### Deleting after a value:

Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 10  
    [END OF IF]

Step 2: SET PTR = START

Step 3: SET PREPTR = PTR

Step 4: Repeat Step 5 and 6 while PREPTR->DATA != NUM

Step 5: SET PREPTR = PTR

Step 6: SET PTR = PTR->NEXT

    [END OF LOOP]

Step 7: SET TEMP = PTR

Step 8: SET PREPTR->NEXT = TEMP->NEXT

Step 9: FREE TEMP

Step 10: EXIT

\*\*\*

### Searching a linked list:

Step 1: [INITIALIZE] SET PTR = START

Step 2: Repeat Step 3 while PTR != NULL

Step 3: IF VAL = PTR->DATA  
    PRINT 'ELEMENT FOUND'  
    Go To Step 5  
ELSE  
    SET PTR = PTR->NEXT  
[END OF IF]  
  
[END OF LOOP]

Step 4: PRINT 'ELEMENT NOT FOUND' //if search unsuccessful

Step 5: EXIT

### Displaying elements of linked list:

Step 1: [INITIALIZE] SET PTR = START

Step 2: Repeat Steps 3 and 4 while PTR != NULL

Step 3: PRINT PTR->DATA

Step 4: SET PTR = PTR->NEXT

[END OF LOOP]

Step 5: EXIT

\*\*\*

### Sorting linked list:

1.Create function SortList which has two attributes: head and tail.

2.addNode() will add a new node to the list:

1.Create a new node.

2.It first checks, whether the head is equal to null which means the list is empty.

3.If the list is empty, both head and tail will point to a newly added node.

4.If the list is not empty, the new node will be added to end of the list such that tail's next will point to a newly added node. This new node will become the new tail of the list.

3.sortList() will sort the nodes of the list in ascending order.

1.Define a node current which will point to head.

2.Define another node index which will point to node next to current.

3.Compare data of current and index node. If current's data is greater than the index's data then, swap the data between them.

4.Current will point to current.next and index will point to index.next.

5.Continue this process until the entire list is sorted.

4.display() will display the nodes present in the list:

1.Define a node current which will initially point to the head of the list.

2.Traverse through the list till current points to null.

3.Display each node by making current to point to node next to it in each iteration.

### Reversing a linked list:

Step 1: Initialize three pointers prev as NULL, curr as head and next as NULL.

Step 2: Iterate through the linked list. In loop, do following.

```
// Before changing next of current,  
// store next node  
    next = curr->next  
// Now change next of current  
// This is where actual reversing happens  
    curr->next = prev  
  
// Move prev and curr one step forward  
    prev = curr  
    curr = next
```

### Updating first node:

Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 5  
[END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START->NEXT

Step 4: SET PTR->DATA = VAL

Step 5: EXIT

### Updating last node:



Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 8  
[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Steps 4 and 5 while PTR->NEXT != NULL

Step 4: SET PREPTR = PTR

Step 5: SET PTR = PTR->NEXT

[END OF LOOP]

Step 6: SET PREPTR->DATA = VAL

Step 7: EXIT

#### Updating after a value:

Step 1: IF START = NULL, then  
    Write UNDERFLOW  
    Go to Step 10  
[END OF IF]

Step 2: SET PTR = START

Step 3: SET PREPTR = PTR

Step 4: Repeat Step 5 and 6 while PREPTR->DATA != NUM

Step 5: SET PREPTR = PTR

Step 6: SET PTR = PTR->NEXT

[END OF LOOP]

Step 7: SET TEMP = PTR

Step 8: SET PREPTR->NEXT = TEMP->NEXT

Step 9: SET PREPTR->DATA = VAL

Step 10: EXIT