Unit – 3 (Part-1)

Linked List

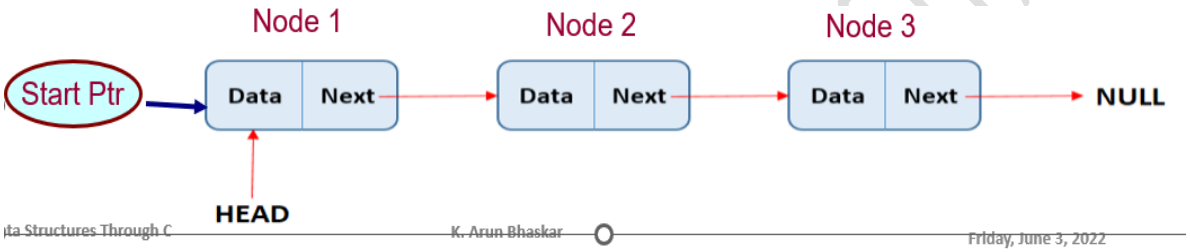
Content

Linked Lists : Introduction, Singly linked list, Operations on Singly Linked list - Insertion, Deletion and Searching.

---

1. Linked lists introduction

- ✓ A linked list, is a linear collection of data elements.
- ✓ These data elements are called nodes.
- ✓ Elements in a linked list can be accessed only in a sequential **manner.**
- ✓ A linked list can be perceived as a train or a sequence of nodes **in** which each node contains one or more data fields and a pointer to the next node.



- ✓ Storing data items in arrays has at least two limitations
  - The array size is fixed once it is created: Changing the size of the array requires creating a new array and then copying all data from the old array to the new array.
  - The data items in the array are next to each other in memory: Inserting an item inside the array requires shifting other items.
- ✓ A linked structure is introduced to overcome limitations of arrays and allow easy insertion and deletion.

Linked Lists:

- ✓ A collection of nodes storing data items and links to other nodes
- ✓ If each node has a data field and a reference field to another node called next or successor, the sequence of nodes is referred to as a singly linked list
- ✓ Nodes can be located anywhere in the memory
- ✓ The first node is called head and the last node is called tail.
- ✓ In C, we can implement a linked list using the following code:

```
struct node
{
    int data;
    struct node *next;
};
```

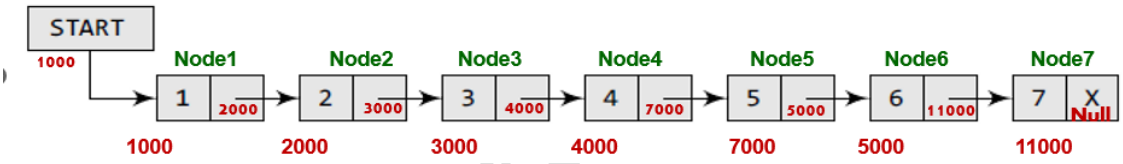
- ✓ Lists contain a pointer variable START that stores the address of the first node in the list.
- ✓ We can traverse the entire list using START which contains the address of the first node; the next part of the first node in turn stores the address of its succeeding node.
- ✓ Using this technique, the individual nodes of the list will form a chain of nodes.
- ✓ If START = NULL, then the linked list is empty and contains no nodes.

Types of Linked lists:

- Singly linked list
- Doubly linked list
- Circular linked list

Single Linked lists introduction:

- ✓ A singly linked list is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type.
- ✓ By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next node in sequence.
- ✓ A singly linked list allows traversal of data only in one way.



Single Linked lists Operations

- ✓ Traversing the linked list
- ✓ Searching an element
- ✓ Insert an element.
- ✓ Delete an element

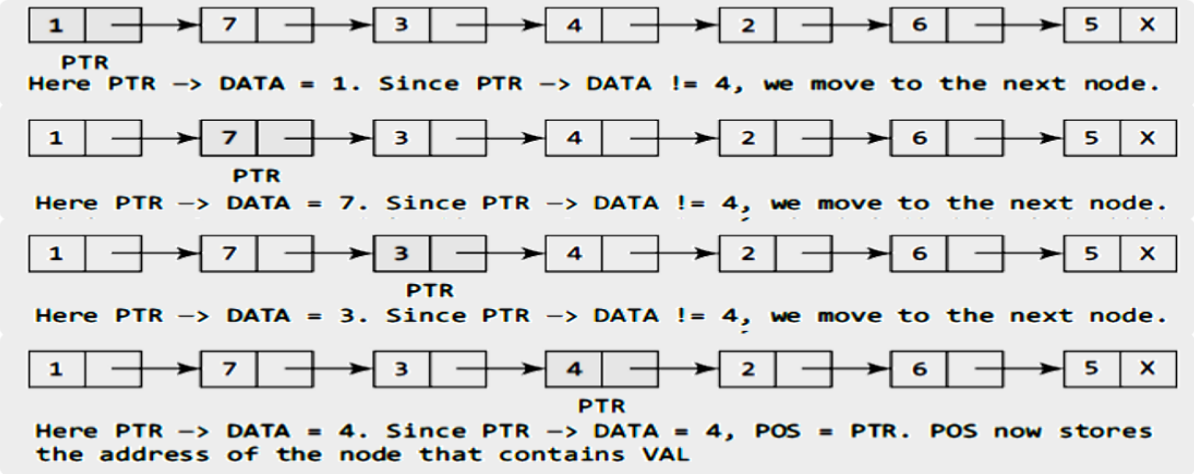
Algorithm to traversing a Linked List:

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3:          Apply Process to PTR->DATA
Step 4:          SET PTR = PTR->NEXT
      [END OF LOOP]
Step 5: EXIT
```

Algorithm to search a value in a Linked List:

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Step 3 while PTR != NULL
Step 3:      IF VAL = PTR -> DATA
                   SET POS = PTR
                   Go To Step 5
             ELSE
                   SET PTR = PTR -> NEXT
             [END OF IF]
        [END OF LOOP]
Step 4: SET POS = NULL
Step 5: EXIT
```

Example: to search the node data = 4:



Algorithm to count the number of nodes in a Linked List :

```
Step 1: [INITIALIZE] SET COUNT = 0
Step 2: [INITIALIZE] SET PTR = START
Step 3: Repeat Steps 4 and 5 while PTR != NULL
Step 4:          SET COUNT = COUNT + 1
Step 5:          SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 6: Write COUNT
Step 7: EXIT
```

Inserting a New Node in a Linked List

- ✓ **New node is added into an already existing linked list.**
- ✓ **Four different cases of insertion in Linked Lists:**
- **Case 1: The new node is inserted at the beginning.**
- **Case 2: The new node is inserted at the end.**
- **Case 3: The new node is inserted after a given node.**
- **Case 4: The new node is inserted before a given node.**

Inserting a Node at the Beginning of a Linked List

✓ Suppose we want to add a new node with data 9 and add it as the first node of the list, then the following changes will be done in the linked list.

Step 1: IF PTR = NULL

Write OVERFLOW
Go to Step 7
[END OF IF]

Step 2: SET NEW_NODE = PTR

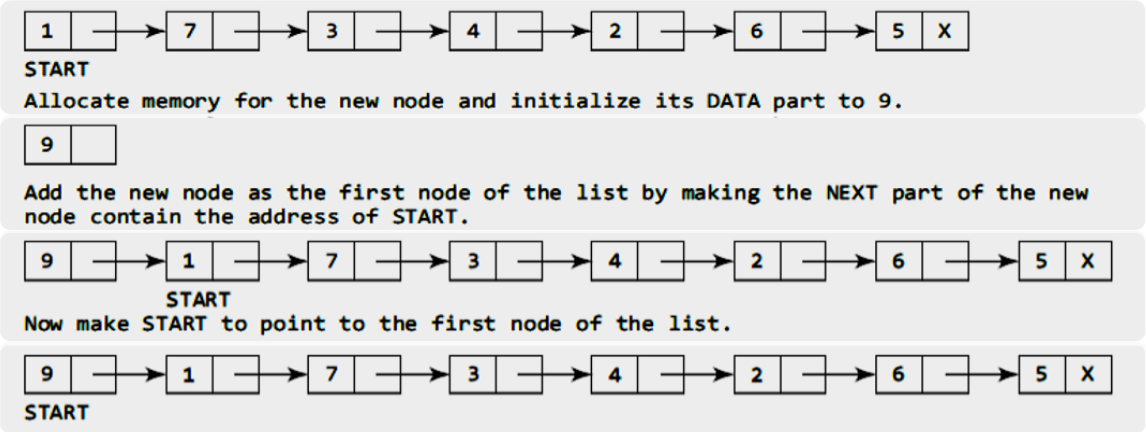
Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW_NODE → DATA = VAL

Step 5: SET NEW_NODE → NEXT = HEAD

Step 6: SET HEAD = NEW_NODE

Step 7: EXIT



Inserting a Node at the End of a Linked List

Step 1: IF PTR = NULL

Write OVERFLOW
Go to Step 7
[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW_NODE → DATA = VAL

Step 5: SET NEW_NODE → NEXT = NULL

Step 6: SET PTR = START

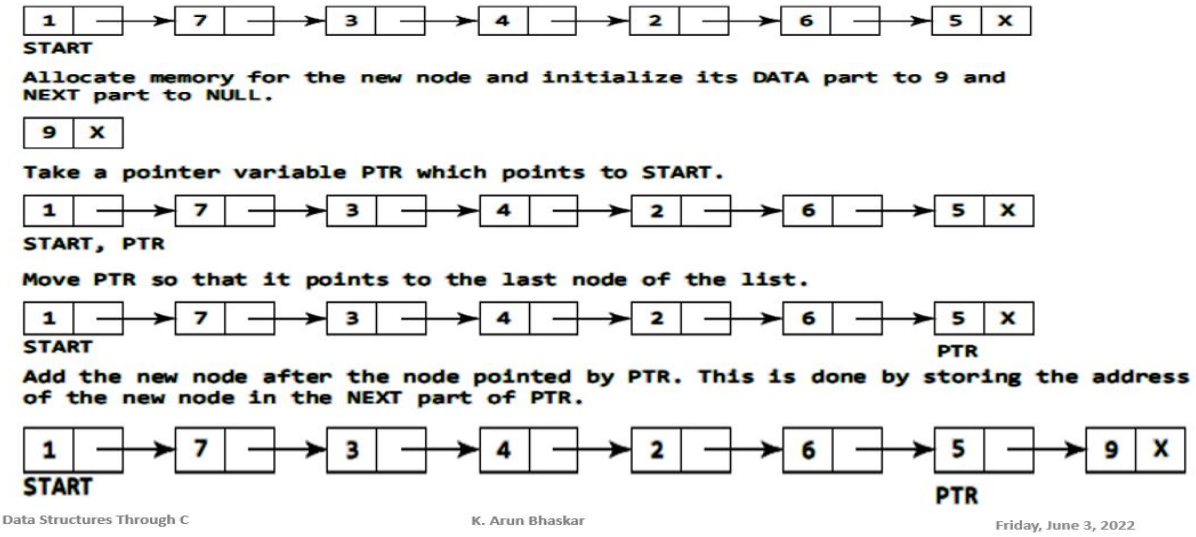
Step 7: Repeat step 8 while PTR->NEXT != NULL

Step 8: SET PTR = PTR → NEXT

[END OF LOOP]

Step 9: SET PTR →NEXT=NEW_NODE

Step 10: EXIT



Inserting a Node After a Given Node in a Linked List

- Suppose we want to add a new node with value 9 after the node containing data 3.

Step 1: IF PTR = NULL

Write OVERFLOW
Go to Step 12
[END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW_NODE → DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat step 8 and 9 while PREPTR → DATA != NUM

Step 8: SET PREPTR = PTR

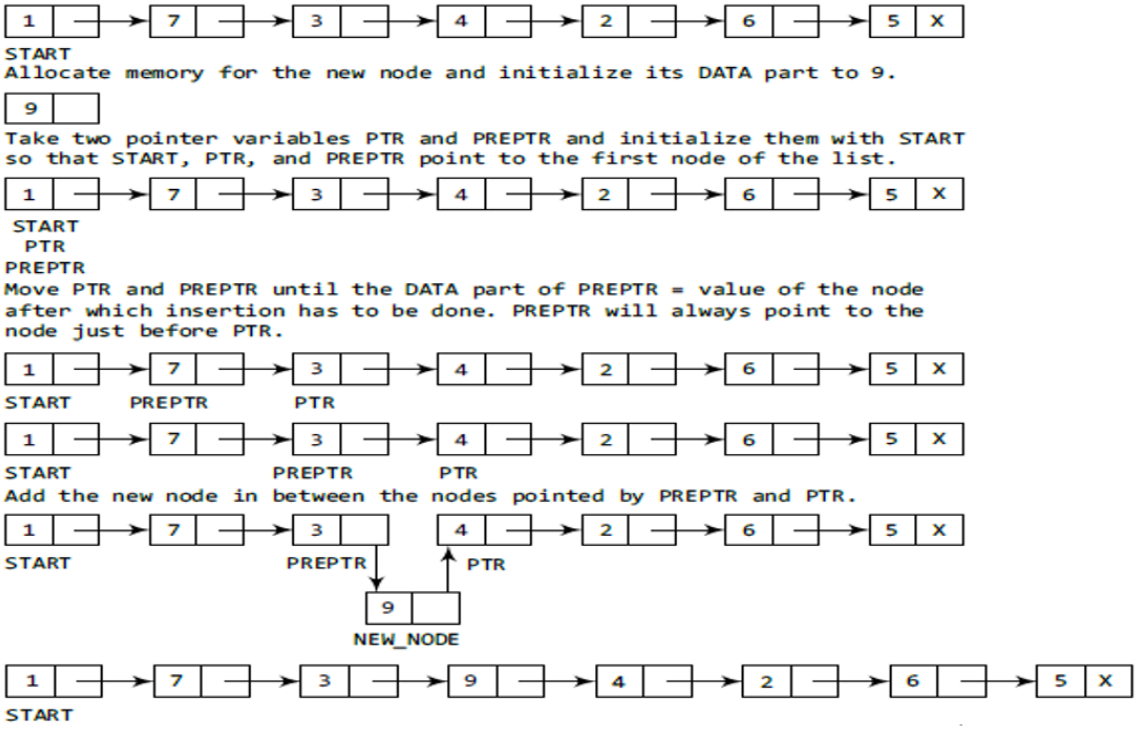
Step 9: PTR = PTR→ NEXT

[END OF LOOP]

Step 10: PREPTR →NEXT=NEW_NODE

Step 11: SET NEW_NODE →NEXT = PTR

Step 12: EXIT



Inserting a Node Before a Given Node in a Linked List

- **Suppose we want to add a new node with value 9 before the node containing 3**
- **Step 1:** IF PTR = NULL

  Write OVERFLOW
  Go to Step 12
  [END OF IF]

Step 2: SET NEW_NODE = PTR

Step 3: SET PTR = PTR → NEXT

Step 4: SET NEW_NODE → DATA = VAL

Step 5: SET PTR = START

Step 6: SET PREPTR = PTR

Step 7: Repeat step 8 and 9 while PTR → DATA != NUM

Step 8: SET PREPTR = PTR

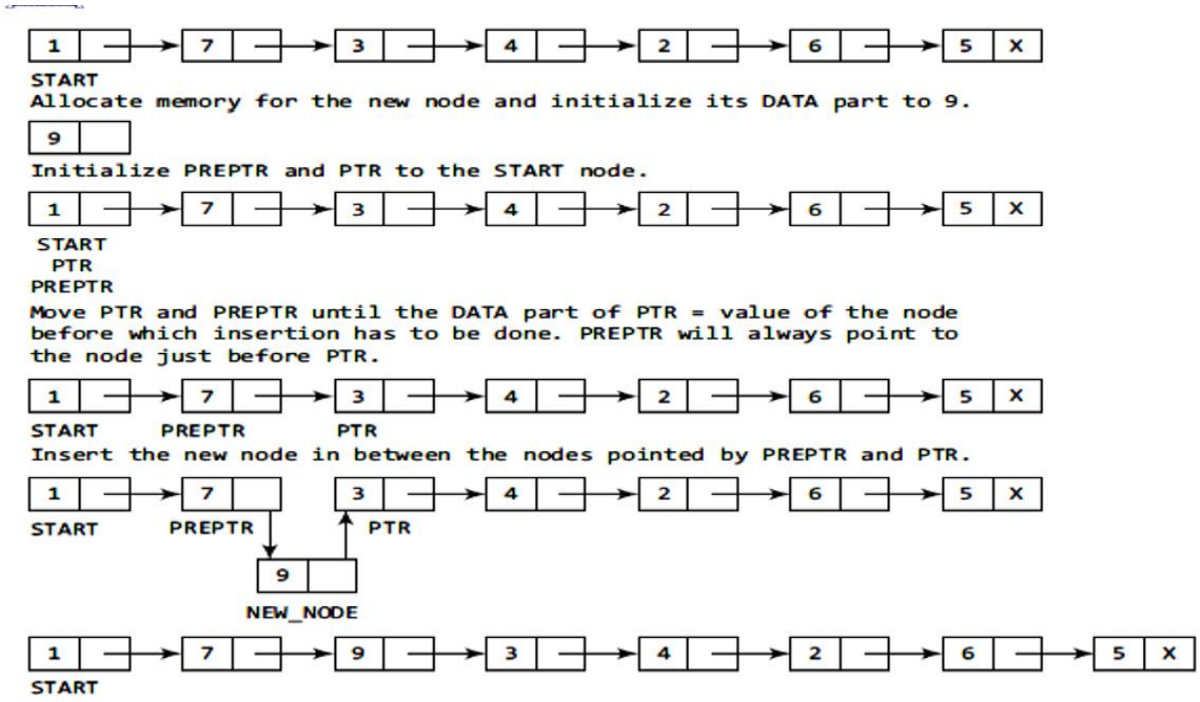
Step 9: SET PTR = PTR →NEXT

[END OF LOOP]

Step 10: PREPTR →NEXT=NEW_NODE

Step 11: SET NEW_NODE →NEXT = PTR

Step 12: EXIT



Deleting a Node from a Linked List

A node is deleted from an already existing linked list with three cases:

- Case 1: The first node is deleted.
- Case 2: The last node is deleted.
- Case 3: The node after a given node is deleted.

Before we describe the algorithms in all these three cases:

✓ First discuss an important term called UNDERFLOW.

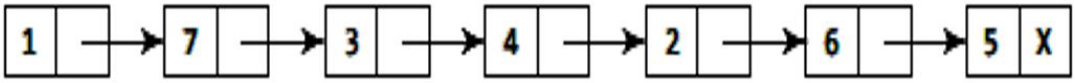
✓ Underflow is a condition that occurs when we try to delete a node from a linked list that is empty.

✓ This happens when START = NULL or when there are no more nodes to delete.

Deleting the First Node from a Linked List

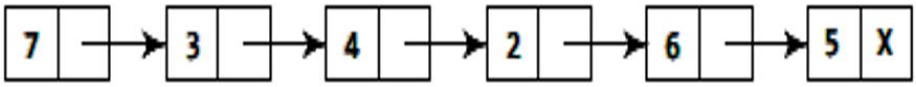
- When we want to delete a node from the beginning of the list, then the following changes will be done in the linked list.

Step 1: IF START = NULL
Write OVERFLOW
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



START

Make START to point to the next node in sequence.

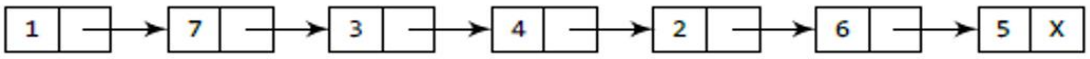


START

Deleting the Last Node from a Linked List

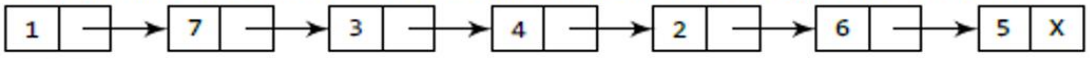
- Suppose we want to delete the last node from the linked list, then the following changes will be done in the linked list.

```
Step 1: IF START = NULL
            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
```



START

Take pointer variables PTR and PREPTR which initially point to START.

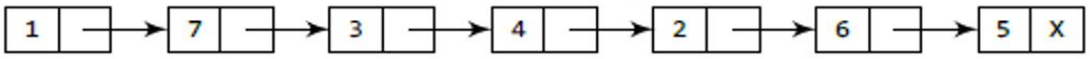


START

PREPTR

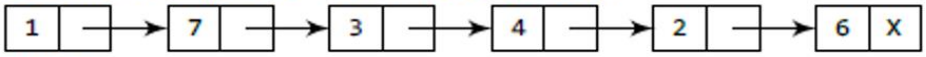
PTR

Move PTR and PREPTR such that NEXT part of PTR = NULL. PREPTR always points to the node just before the node pointed by PTR.



START PREPTR PTR

Set the NEXT part of PREPTR node to NULL.

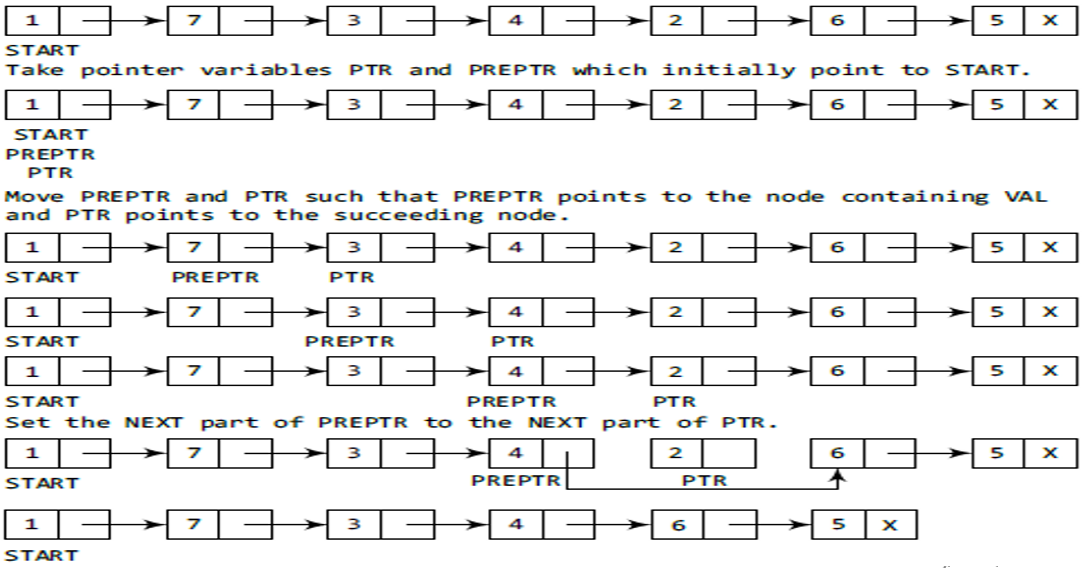


START

Deleting the Node After a Given Node in a Linked List

- Suppose we want to delete the node that succeeds the node which contains data value 4. Then the following changes will be done in the linked list.

```
Step 1: IF START = NULL
            Write UNDERFLOW
            Go to Step 10
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET PREPTR = PTR
Step 4: Repeat Steps 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 = PTR -> NEXT
Step 9: FREE TEMP
Step 10: EXIT
```



Single Linked lists Operations Summary

- ✓ Traversing the linked list
- ✓ Searching an element
- ✓ Counting nodes in a list
- ✓ Insert an element.
  - Case 1: *The new node is inserted at the beginning.*
  - Case 2: *The new node is inserted at the end.*
  - Case 3: *The new node is inserted after a given node.*
  - Case 4: *The new node is inserted before a given node.*
- ✓ Delete an element
  - Case 1: *The first node is deleted.*
  - Case 2: *The last node is deleted.*
  - Case 3: *The node after a given node is deleted.*

Applications of Linked Lists

- ✓ Linked lists can be used to represent polynomials and the different operations that can be performed on them.
- ✓ **Polynomial Representation:**
- ✓ Consider a polynomial $6x^3+9x^2+7x+1$.
- ✓ Every individual term in a polynomial consists of two parts, a coefficient and a power.
- ✓ Here, 6, 9, 7, and 1 are the coefficients of the terms that have 3, 2, 1, and 0 as their powers respectively.
- ✓ Every term of a polynomial can be represented as a node of the linked list.
- ✓ Figure shows the linked representation of the terms of the above polynomial.

***