Data Structure-Linked List

Kaustuv Bhattacharjee

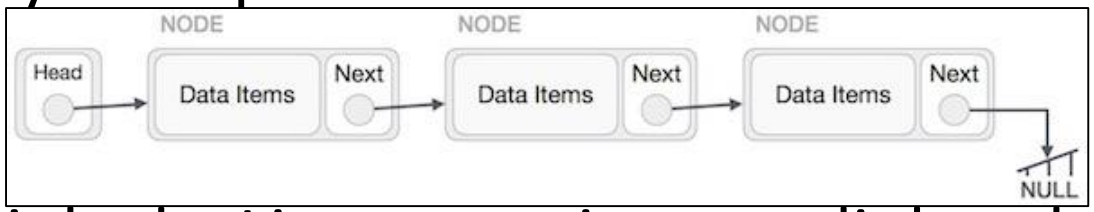
University of Engineering & Management, Kolkata

Linked List-Introduction

- A sequence of data elements, which are connected together via links
- Data elements are called *nodes*
- Each node contains a connection to another node

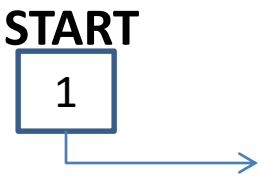
Linked List-Representation

- Linked list can be visualized as a chain of nodes, where every node points to the next node



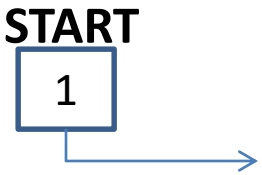
  - Linked List contains a link element called *first/start/head*
  - Each node carries a *data* field(s) and a link field called *next*
  - Each node is linked with its next node using its *next* link
  - Last node carries a link as *null* to mark the end of the list

Linked List-Memory Representation



	Data	Next
1	75	4
2	21	7
3		
4	85	2
5		
6		
7	56	-1

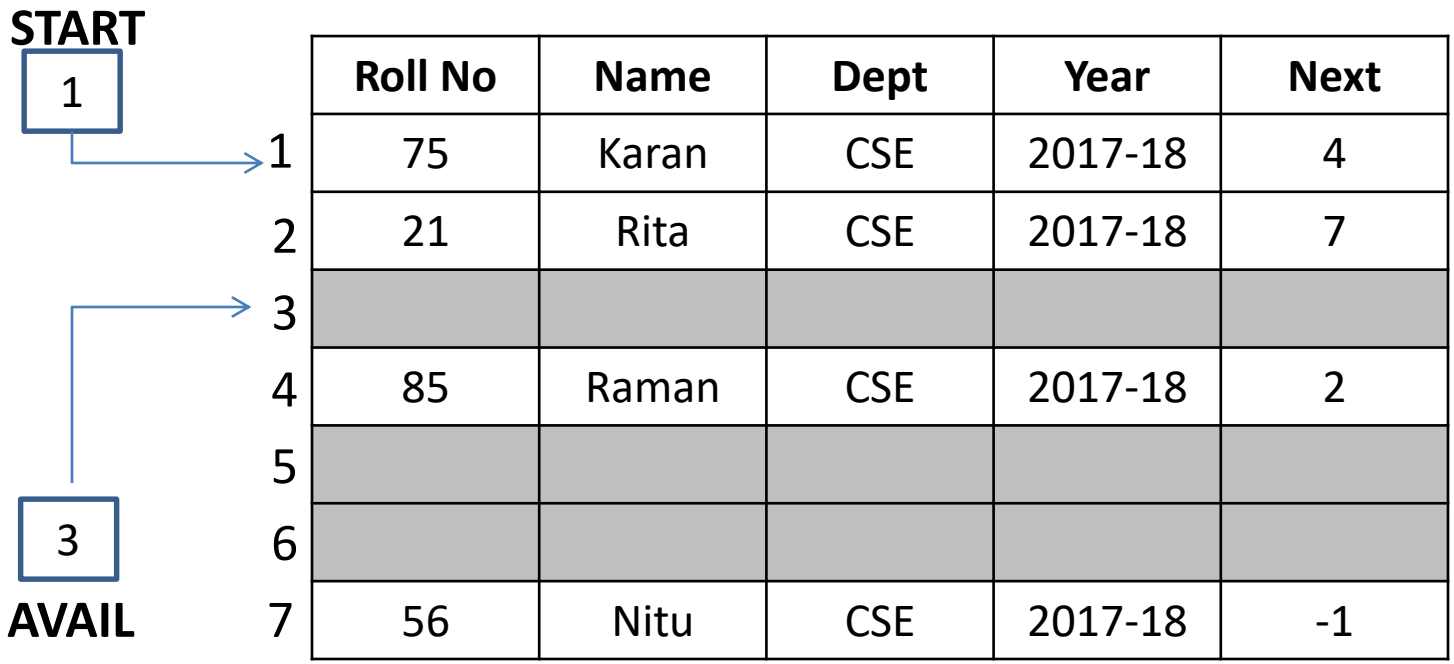
Linked List-Structure as Data elements



	Roll No	Name	Dept	Year	Next
1	75	Karan	CSE	2017-18	4
2	21	Rita	CSE	2017-18	7
3					
4	85	Raman	CSE	2017-18	2
5					
6					
7	56	Nitu	CSE	2017-18	-1

Representation of a structure using a linked list

Linked List-Memory Allocation and De-allocation



	Roll No	Name	Dept	Year	Next
1	75	Karan	CSE	2017-18	4
2	21	Rita	CSE	2017-18	7
3					
4	85	Raman	CSE	2017-18	2
5					
6					
7	56	Nitu	CSE	2017-18	-1

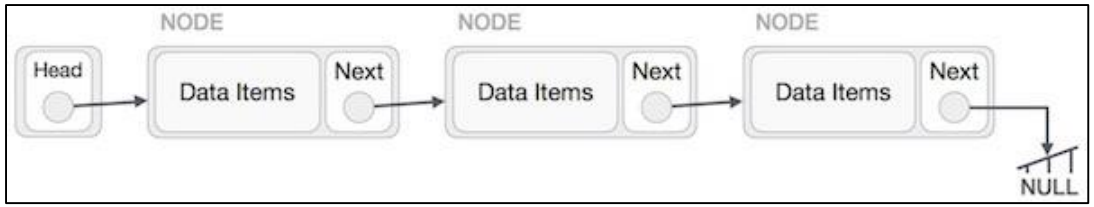
Linked List with **START** and **AVAIL** pointers

Linked List-Types

- **Simple Linked List** – Item navigation is forward only
- **Doubly Linked List** – Items can be navigated forward and backward
- **Circular Linked List** – Last item contains link of the first element as next and the first element has a link to the last element as previous

Simple Linked List

- **Simple (Singly) Linked List**
  - Every node contains data and pointer to the next node of same data type
  - Only one way data traversal (forward)



Single Linked List-Basic Operations

- **Count –** Count number of nodes in a linked list
- **Display /Traverse**– Displays the complete list
- **Search** – Search for a value in the linked list
- **Insertion** – Insert a node *at the beginning /at the end/after a given node/before a given node* of the list.
- **Deletion** – Delete a node : *the First node/ the Last node /after a given node/before a given node* of the list.

Single Linked List-Count Nodes

- Steps/Algorithm:

  Step 1: [Initialization] Set COUNT=0

  Step 2: [Initialization] 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

  [loop ends]

  Step 6: Write COUNT

  Step 7: Stop

Single Linked List-Traversal

- Steps/Algorithm:

  Step 1: [Initialization] 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

  [loop ends]

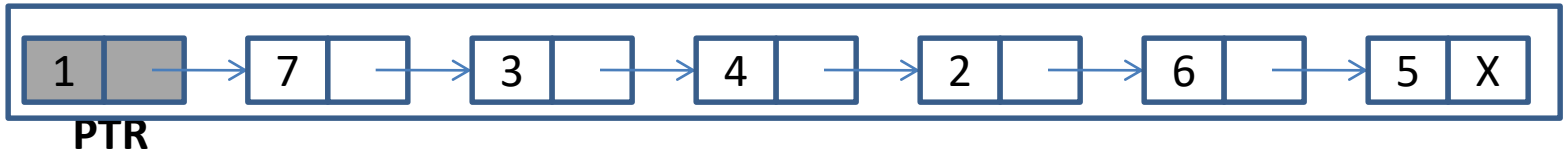
  Step 5: Stop

Single Linked List-Search

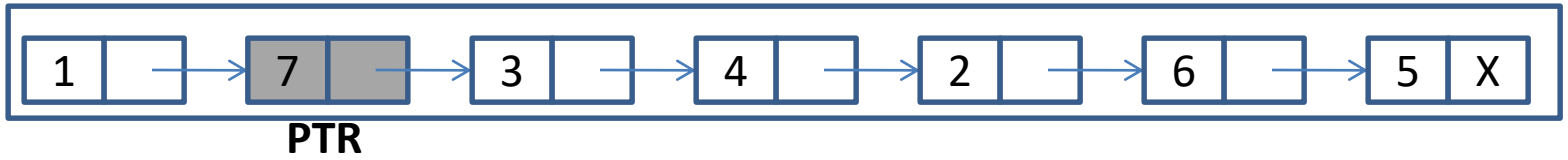
- Steps/Algorithm:

```
Step 1: [Initialization] 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]
                [loop ends]
Step 4:         Set POS=NULL
Step 5: Stop
```

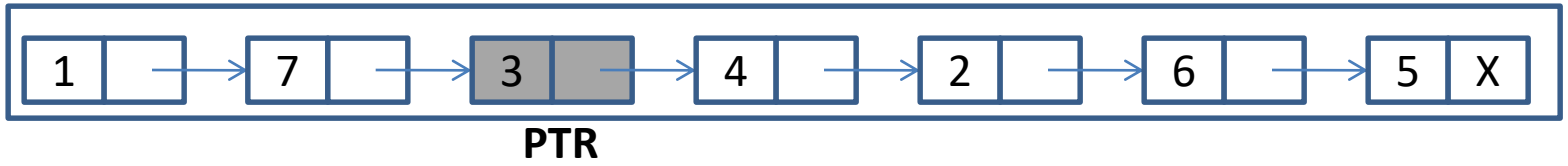
Single Linked List-Search Explanation



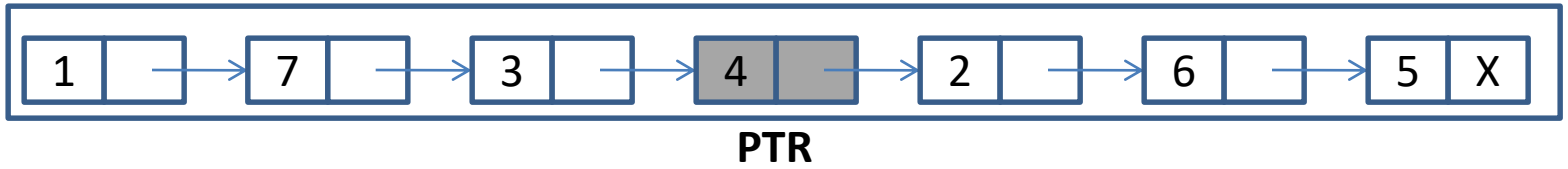
Here PTR->DATA=1. Since PTR->DATA!=4, move to next node



Here PTR->DATA=7. Since PTR->DATA!=4, move to next node



Here PTR->DATA=3. Since PTR->DATA!=4, move to next node



Here PTR->DATA=4. Since PTR->DATA=4, POS=PTR.

Single Linked List-Insertion

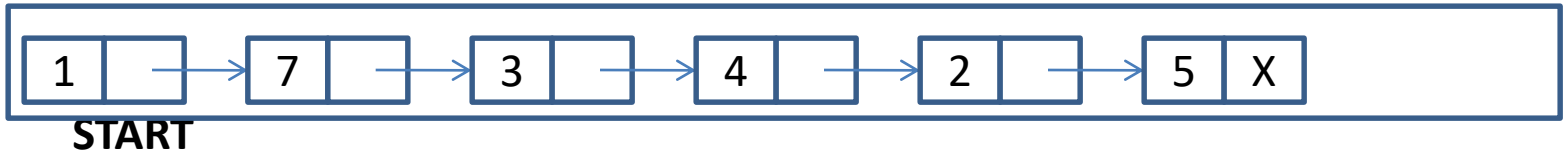
- Various scenarios of Inserting a node in Linked List:
  - Insert a new Node at the beginning of a Linked List
  - Insert a new Node at the end of a Linked List
  - Insert a new Node after a node in a Linked List
  - Insert a new Node before a node in a Linked List

Single Linked List-Insertion Contd…

- Steps/Algorithm [Insert at beginning]:

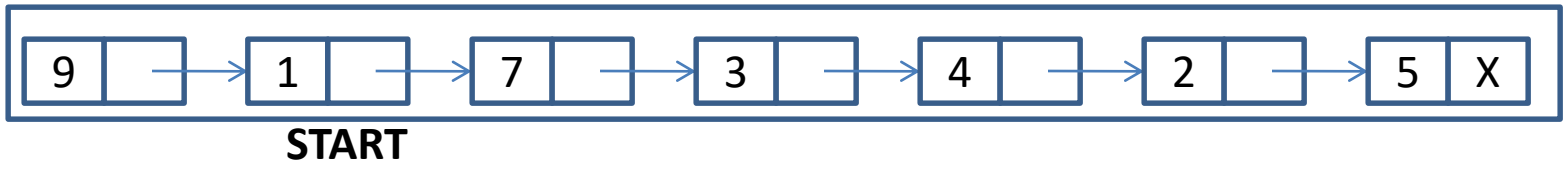
Step 1: If AVAIL=NULL
Write OVERFLOW
GO TO Step 7
[End of IF]
Step 2: Set NEW_NODE=AVAIL
Step 3: Set AVAIL=AVAIL->NEXT
Step 4: Set NEW_NODE->DATA= VAL
Step 5: Set NEW_NODE->NEXT= START
Step 6: Set START=NEW_NODE
Step 7: Stop

Single Linked List-Insert Explanation

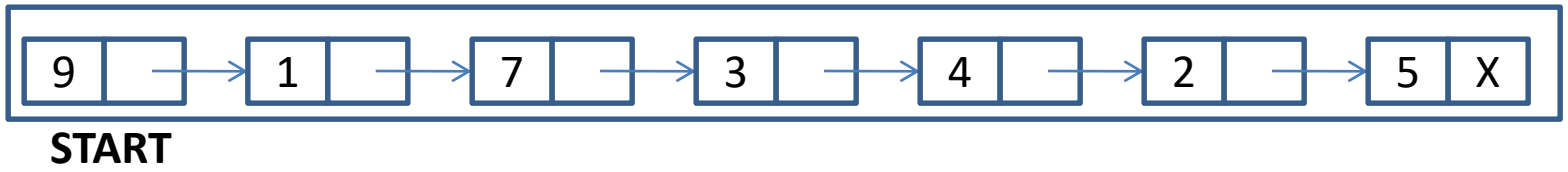


Allocate memory for new node and initialize its DATA to 9

Add the new node as the first node of the list making the NEXT part of the new node contain the address of START.



Now make START to point to the first node of the list

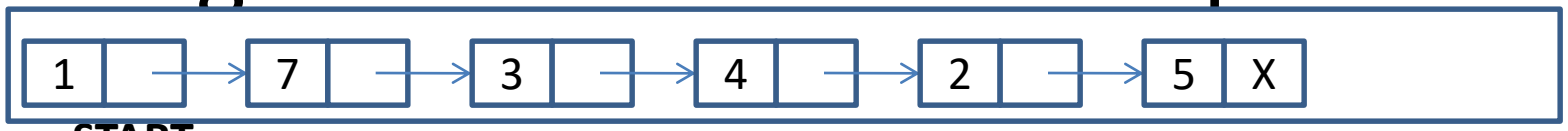


Single Linked List-Insertion Contd…

- Steps/Algorithm [Insert at End]:

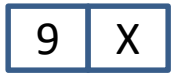
```
Step 1: If AVAIL=NULL
                Write OVERFLOW
                GO TO Step 10
            [End of IF]
Step 2: Set NEW_NODE=AVAIL
Step 3: Set AVAIL=AVAIL->NEXT
Step 4: Set NEW_NODE->DATA= VAL
Step 5: Set NEW_NODE->NEXT= NULL
Step 6: Set PTR= START
Step 7: Repeat step 8 while PTR->NEXT!=NULL
Step 8:         Set PTR=PTR->NEXT
          [Loop ends]
Step 9: Set PTR->NEXT=NEW_NODE
Step 10: Stop
```

Single Linked List-Insert Explanation

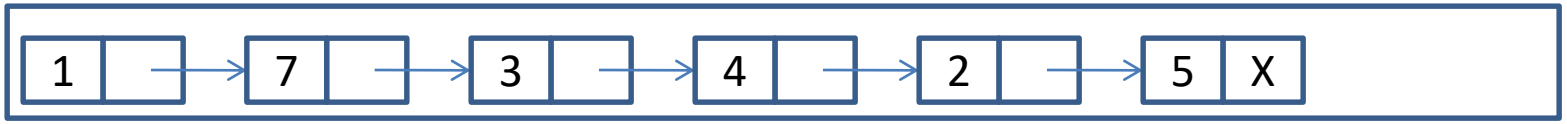


START

Allocate memory for new node and initialize its DATA to 9 and NEXT part to NULL.

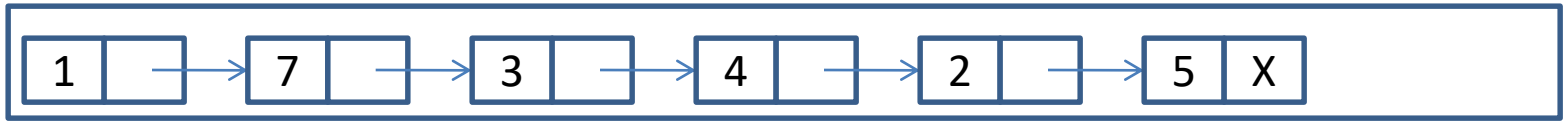


Take a pointer variable PTR which points to START.



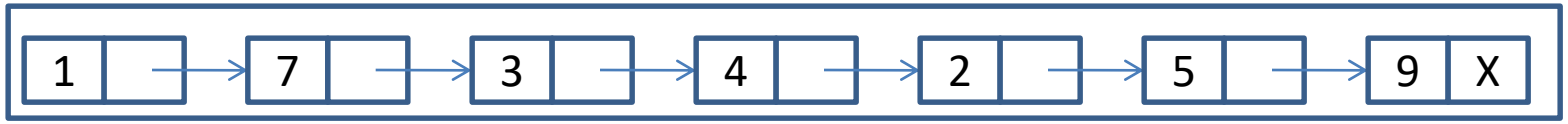
START,PTR

Move PTR so that it points to the last node of the list.



START **PTR**

Add the new node after the last node pointer by PTR; NEXT node of PTR should point to the new node.



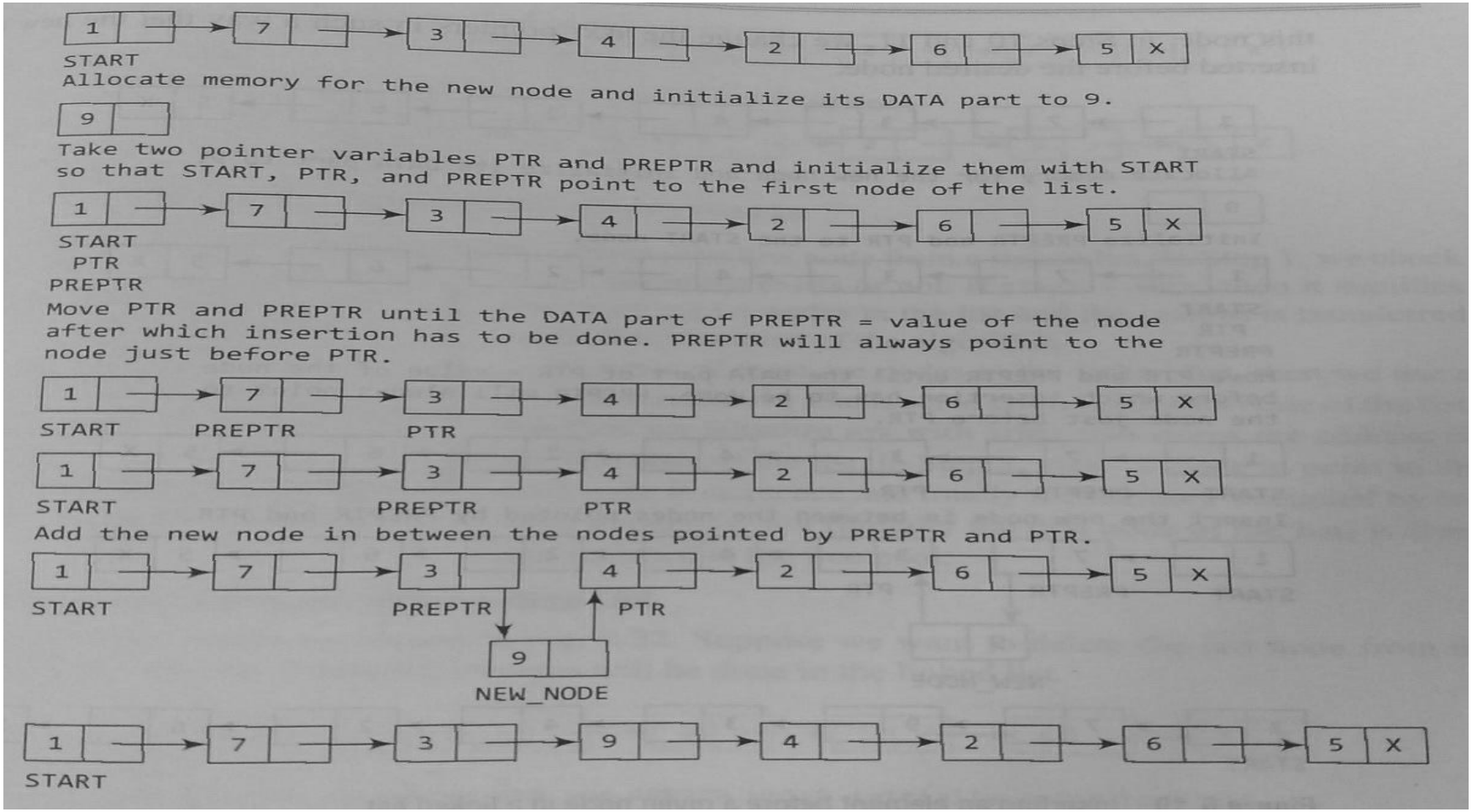
START **PTR**

Single Linked List-Insertion Contd…

- Steps/Algorithm [Insert after a node]:

```
Step 1: If AVAIL=NULL
                    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
            [Loop ends]
Step 10: Set PREPTR>NEXT= NEW_NODE
Step 11: Set NEW_NODE ->NEXT= PTR
Step 12: Stop
```

Single Linked List-Insertion Explanation

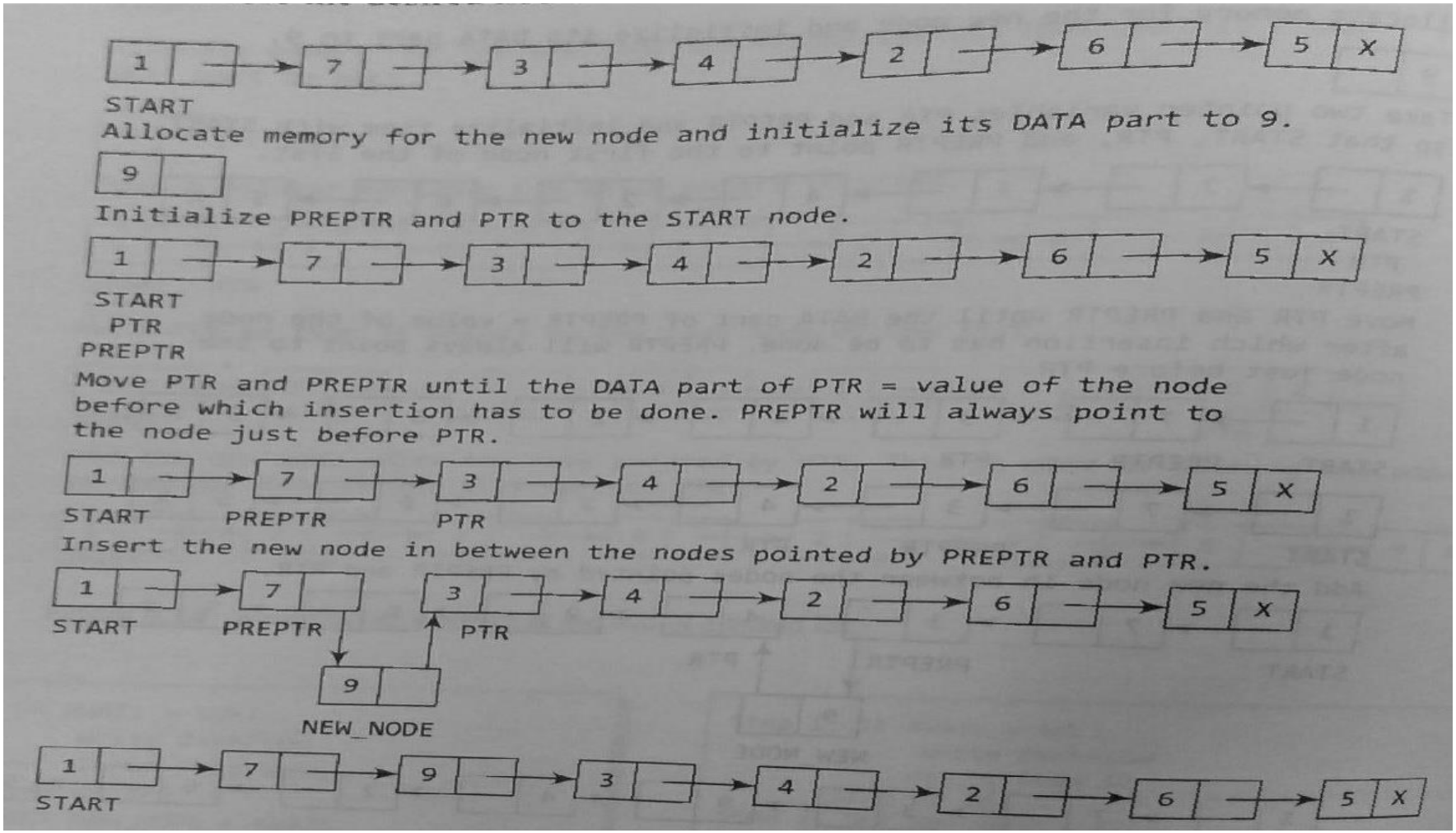


Single Linked List-Insertion Contd…

- Steps/Algorithm [Insert before a node]:

```
Step 1: If AVAIL=NULL
                    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
            [Loop ends]
Step 10: Set PREPTR>NEXT= NEW_NODE
Step 11: Set NEW_NODE ->NEXT= PTR
Step 12: Stop
```

Single Linked List-Insertion Explanation



Single Linked List-Deletion

- Various scenarios of Deleting a node in Linked List:
  - First node of a Linked List is deleted
  - Last node of a Linked List is deleted
  - Node after a node in a Linked List is deleted

Single Linked List-Deletion Contd…

- Steps/Algorithm [Deleting at beginning]:

  Step 1: If START=NULL

  Write UNDERFLOW

  GO TO Step 5

  [End of IF]

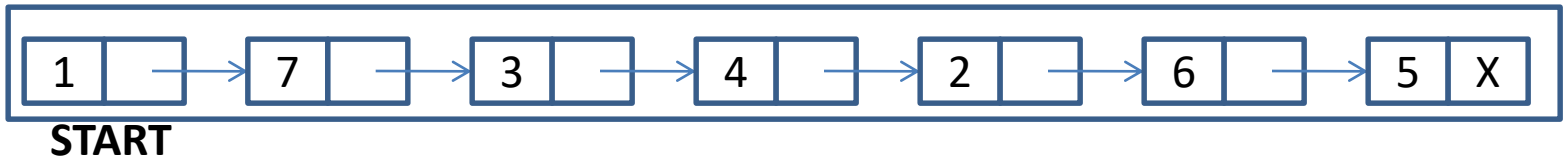
  Step 2: Set PTR=START

  Step 3: Set START=START->NEXT

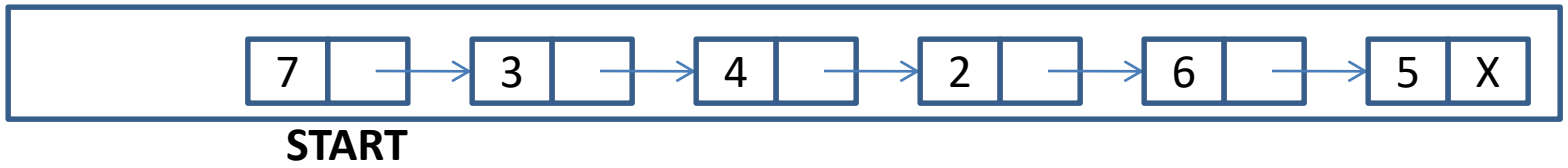
  Step 4: FREE PTR

  Step 5: Stop

Single Linked List-Deletion Explanation



Make START to point to the next node in the sequence.

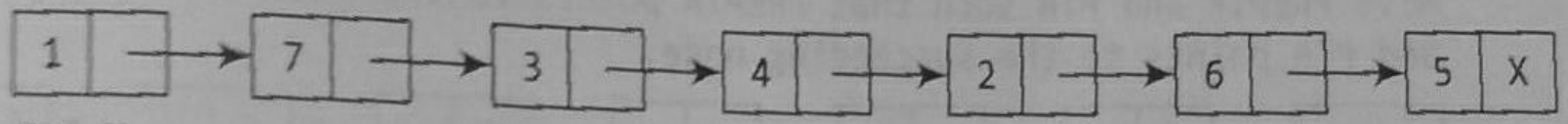


Single Linked List-Deletion Contd…

- Steps/Algorithm [Deleting last node]:

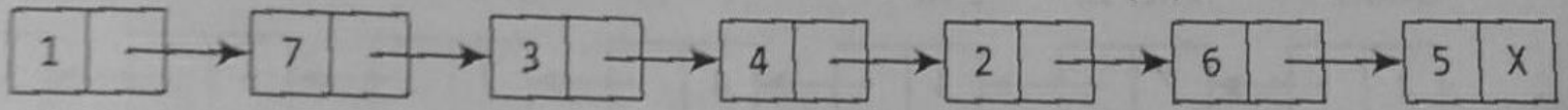
  Step 1: If START=NULL
  Write UNDERFLOW
  GO TO Step 8
  [End of IF]
  Step 2: Set PTR=START
  Step 3: Repeat Steps 4 and 5 while PTR->NEXT !=NULL
  Step 4: Set PREPTR=PTR
  Step 5: Set PTR=PTR->NEXT
  [loop ends]
  Step 6: Set PREPTR->NEXT=NULL
  Step 7: FREE PTR
  Step 8: Stop

Single Linked List-Deletion Explanation



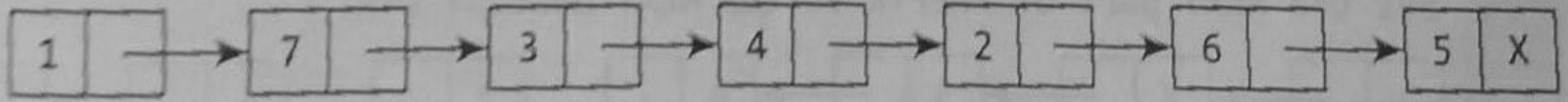
START

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.

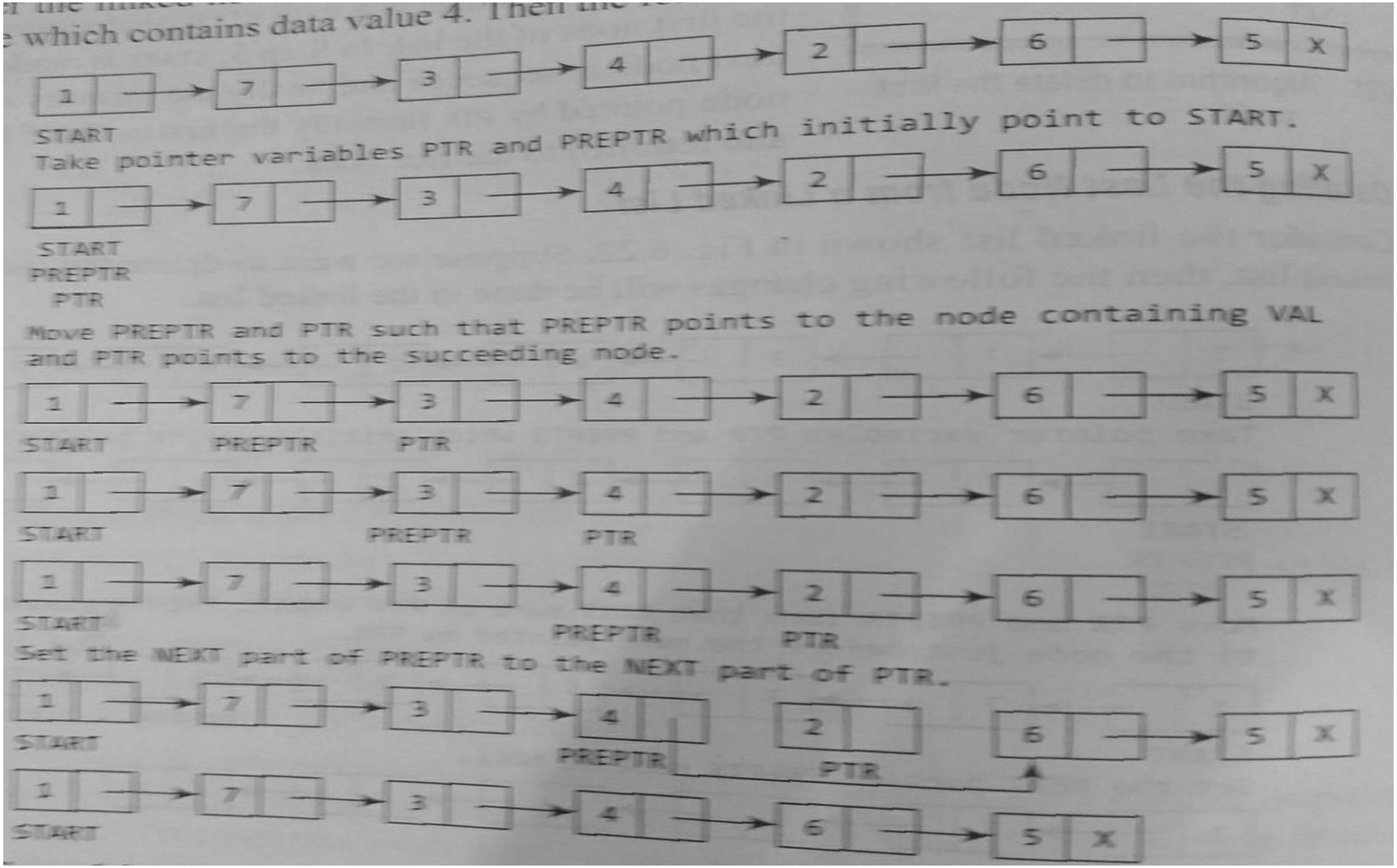
START

Single Linked List-Deletion Contd…

- Steps/Algorithm [Deleting after a given node]:

  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
  [loop ends]
  Step 7: Set TEMP=PTR
  Step 8: Set PREPTR->NEXT=PTR->NEXT
  Step 9: FREE TEMP
  Step 10: Stop

Single Linked List-Deletion Explanation

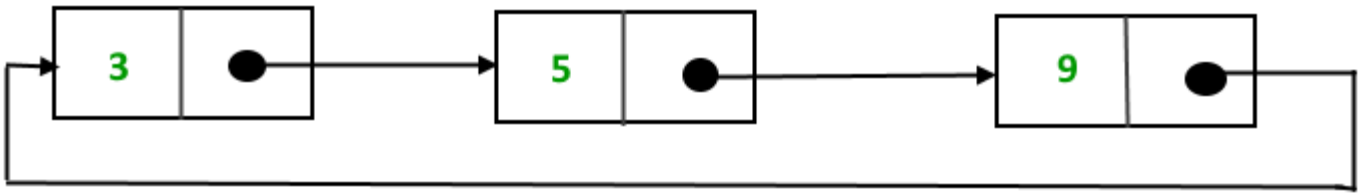


Single Linked List: Assignments

1. Write a program to create a single linked list.
2. Write a program to display a single linked list.
3. Write a program to insert a node at the beginning of a single linked list.
4. Write a program to insert a node at the end of a single linked list.
5. Write a program to insert a node before a given node of a single linked list.
6. Write a program to insert a node after a given node of a single linked list.
7. Write a program to delete a node from the beginning of a single linked list.
8. Write a program to delete a node from the end of a single linked list.
9. Write a program to delete a node after a given node of a single linked list.
10. Write a program to delete a node of a single linked list.
11. Write a program to delete the entire single linked list.

Circular Linked List

- **Circular Linked List**
  - Every node contains data and pointer to the next node of same data type
  - The last node contains the pointer to the first node of the list.
  - Two types: Circular Singly Linked List and Circular Doubly Linked List



Circular Linked List-Basic Operations

- **Insertion** – Insert a node *at the beginning /at the end* of the list.
- **Deletion** – Delete a node : *the First node/ the Last node* of the list.

Circular Linked List-Insertion

- Steps/Algorithm [Insert at beginning]:

  Step 1: If AVAIL=NULL
  Write OVERFLOW
  GO TO Step 11
  [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: Repeat step 7 while PTR->NEXT!=START
  Step 7: Set PTR=PTR->NEXT
  [loop ends]
  Step 8: Set NEW_NODE->NEXT= START
  Step 9: Set PTR->NEXT=NEW_NODE
  Step 10: Set START=NEW_NODE
  Step 11: Stop

Circular Linked List-Insertion Contd…

- Steps/Algorithm [Insert at end]:

  Step 1: If AVAIL=NULL
  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: NEW_NODE->NEXT=START
  Step 6: Set PTR=START
  Step 7: Repeat step 8 while PTR->NEXT!=START
  Step 8: Set PTR=PTR->NEXT
  [loop ends]
  Step 9: Set PTR->NEXT=NEW_NODE
  Step 10: Stop

Circular Linked List-Deletion

- Steps/Algorithm [Deleting at beginning]:

Step 1: If START=NULL
Write UNDERFLOW
GO TO Step 8
[End of IF]
Step 2: Set PTR=START
Step 3: Repeat Step 4 while PTR->NEXT!=START
Step 4: Set PTR=PTR->NEXT
[loop ends]
Step 5: Set PTR->NEXT=START->NEXT
Step 6: FREE START
Step 7: Set START=PTR->NEXT
Step 8: Stop

Circular Linked List-Deletion

- Steps/Algorithm [Deleting at end]:

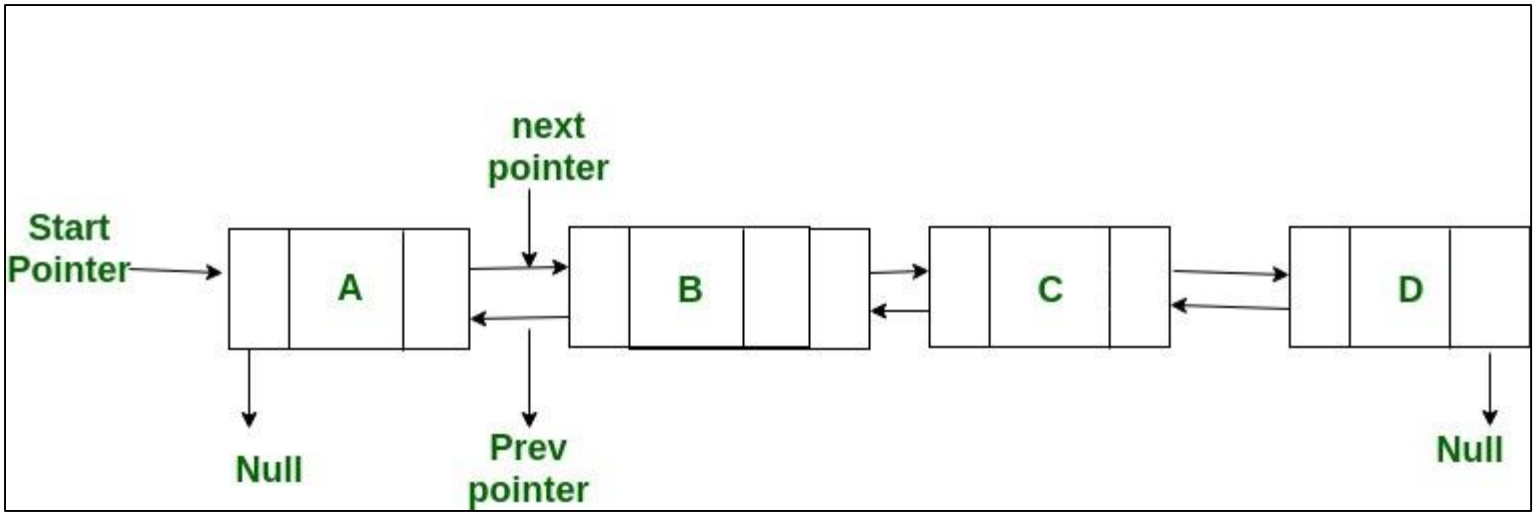
  Step 1: If START=NULL
  Write UNDERFLOW
  GO TO Step 8
  [End of IF]
  Step 2: Set PTR=START
  Step 3: Repeat Steps 4 and 5 while PTR->NEXT!=START
  Step 4: Set PREPTR=PTR
  Step 5: Set PTR-=PTR->NEXT
  [loop ends]
  Step 6: Set PREPTR->NEXT=START
  Step 7: FREE PTR
  Step 8: Stop

Circular Linked List: Assignments

1. Write a program to create a circular linked list.
2. Write a program to display a circular linked list.
3. Write a program to insert a node at the beginning of a circular linked list.
4. Write a program to insert a node at the end of a circular linked list.
5. Write a program to delete a node from the beginning of a circular linked list.
6. Write a program to delete a node from the end of a circular linked list.
7. Write a program to delete a node after a given node of a circular linked list.
8. Write a program to delete the entire circular linked list.

Doubly Linked List

- **Doubly Linked List**
  - A two way linked list containing a pointer to the next as well as to the previous node in a sequence
  - Every node consists of three parts: data, pointer to the previous node and pointer to the next node



Doubly Linked List-Insertion

- Various scenarios of Inserting a node in Doubly Linked List:
  - Insert a new Node at the beginning of a Doubly Linked List
  - Insert a new Node at the end of a Doubly Linked List
  - Insert a new Node after a node in a Doubly Linked List
  - Insert a new Node before a node in a Doubly Linked List

Doubly Linked List-Insertion Contd…

- Steps/Algorithm [Insert at beginning]:

  Step 1: If AVAIL=NULL
  Write OVERFLOW
  GO TO Step 9
  [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->PREV= NULL
  Step 6: Set NEW_NODE->NEXT= START
  Step 7: Set START->PREV=NEW_NODE
  Step 8: Set START=NEW_NODE
  Step 9: Stop

Doubly Linked List-Insertion Contd…

- Steps/Algorithm [Insert at end]:

```
Step 1: If AVAIL=NULL
                Write OVERFLOW
                GO TO Step 11
            [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
          [loop ends]
Step 9: Set PTR->NEXT= NEW_NODE
Step 10: Set NEW_NODE->PREV=PTR
Step 11: Stop
```

Doubly Linked List-Insertion Contd…

- Steps/Algorithm [Insert after a given node]:

```
Step 1: If AVAIL=NULL
                    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: Repeat  Step 7 while PTR->DATA!=NUM
Step 7:              Set PTR=PTR->NEXT
           [loop ends]
Step 8: Set NEW_NODE->NEXT= PTR->NEXT
Step 9: Set NEW_NODE->PREV=PTR
Step 10: Set PTR->NEXT=NEW_NODE
Step 11: Set PTR>NEXT->PREV= NEW_NODE
Step 12: Stop
```

Doubly Linked List-Insertion Contd…

- Steps/Algorithm [Insert before a given node]:

```
Step 1: If AVAIL=NULL
                    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: Repeat  Step 7 while PTR->DATA!=NUM
Step 7:              Set PTR=PTR->NEXT
            [loop ends]
Step 8: Set NEW_NODE->NEXT= PTR
Step 9: Set NEW_NODE->PREV=PTR->PREV
Step 10: Set PTR->PREV=NEW_NODE
Step 11: Set PTR->PREV->NEXT= NEW_NODE
Step 12: Stop
```

Doubly Linked List-Deletion

- Steps/Algorithm [Delete First node]:

Step 1: If START=NULL
Write UNDERFLOW
GO TO Step 6
[End of IF]
Step 2: Set PTR=START
Step 3: Set START=START->NEXT
Step 4: Set START->PREV=NULL
Step 5: FREE PTR
Step 6: Stop

Doubly Linked List-Deletion

- Steps/Algorithm [Delete last node]:

Step 1: If START=NULL
Write UNDERFLOW
GO TO Step 7
[End of IF]
Step 2: Set PTR=START
Step 3: Repeat Step 4 while PTR->NEXT!=NULL
Step 4: Set PTR=PTR->NEXT
[loop ends]
Step 5: Set PTR->PREV->NEXT=NULL
Step 6: FREE PTR
Step 7: Stop

Doubly Linked List-Deletion

- Steps/Algorithm [Delete after a given node]:

  Step 1: If START=NULL
  Write UNDERFLOW
  GO TO Step 9
  [End of IF]
  Step 2: Set PTR=START
  Step 3: Repeat Step 4 while PTR->DATA!=NUM
  Step 4: Set PTR=PTR->NEXT
  [loop ends]
  Step 5: Set TEMP=PTR->NEXT
  Step 6: Set PTR->NEXT=TEMP->NEXT
  Step 7: Set TEMP->NEXT->PREV=PTR
  Step 8: FREE TEMP
  Step 9: Stop

Doubly Linked List-Deletion

- Steps/Algorithm [Delete before a given node]:

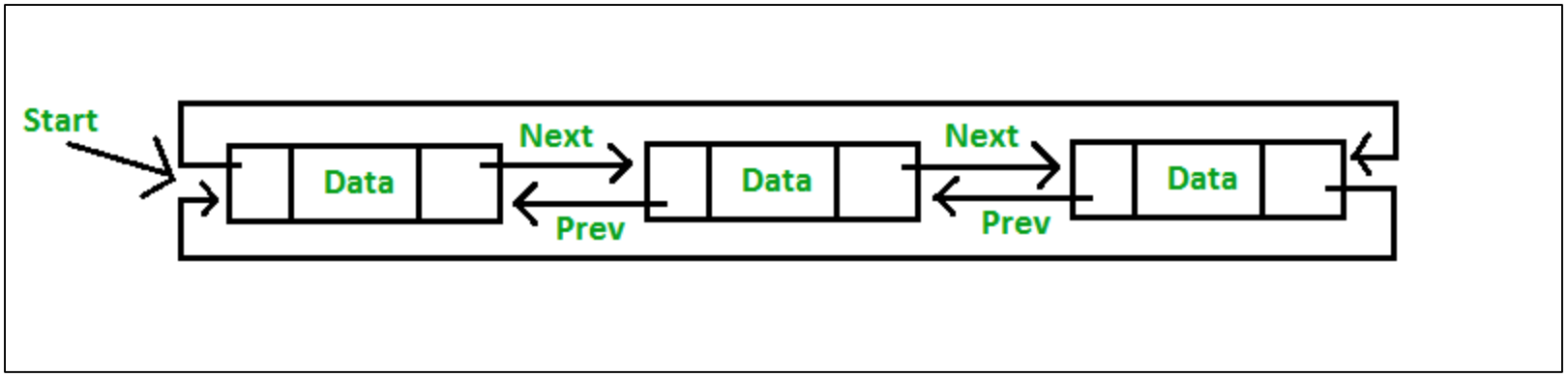
  Step 1: If START=NULL
  Write UNDERFLOW
  GO TO Step 9
  [End of IF]
  Step 2: Set PTR=START
  Step 3: Repeat Step 4 while PTR->DATA!=NUM
  Step 4: Set PTR=PTR->NEXT
  [loop ends]
  Step 5: Set TEMP=PTR->PREV
  Step 6: Set TEMP->PREV->NEXT=PTR
  Step 7: Set PTR->PREV=TEMP->PREV
  Step 8: FREE TEMP
  Step 9: Stop

Doubly Linked List: Assignments

1. Write a program to create a doubly linked list.
2. Write a program to display a doubly linked list.
3. Write a program to insert a node at the beginning of a doubly linked list.
4. Write a program to insert a node at the end of a doubly linked list.
5. Write a program to insert a node before a node of a doubly linked list.
6. Write a program to insert a node after a node of a doubly linked list.
7. Write a program to delete a node from the beginning of a doubly linked list.
8. Write a program to delete a node from the end of a doubly linked list.
9. Write a program to delete a node after a given node of a doubly linked list.
10. Write a program to delete a node before a given node of a doubly linked list.
11. Write a program to delete the entire doubly linked list.

Circular Doubly Linked List

- **Circular Doubly Linked List**
  - A two way linked list containing a pointer to the next as well as to the previous node in a sequence
  - Every node consists of three parts: data, pointer to the previous node and pointer to the next node
  - The last node of the list contains the address of the first node of the list



Circular Doubly Linked List-Basic Operations

- **Insertion** – Insert a node *at the beginning /at the end* of the lists
- **Deletion** – Delete a node : *the First node/ the Last node* of the list

Circular Doubly Linked List-Insertion

- Steps/Algorithm [Insert at beginning]:

```
Step 1: If AVAIL=NULL
                    Write OVERFLOW
                    GO TO Step 13
              [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: Repeat step  7 while PTR->NEXT!=START
Step 7:              Set PTR=PTR->NEXT
           [loop ends]
Step 8: Set PTR->NEXT=NEW_NODE
Step 9: Set NEW_NODE->PREV=PTR
Step 10: Set NEW_NODE->NEXT= START
Step 11: Set START->PREV=NEW_NODE
Step 12: Set START=NEW_NODE
Step 13: Stop
```

Circular Doubly Linked List-Insertion

- Steps/Algorithm [Insert at end]:

  Step 1: If AVAIL=NULL
  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 NEW_NODE->NEXT= START
  Step 6: Set PTR=START
  Step 7: Repeat step 8 while PTR->NEXT!=START
  Step 8: Set PTR=PTR->NEXT
  [loop ends]
  Step 9: Set PTR->NEXT=NEW_NODE
  Step 10: Set NEW_NODE->PREV=PTR
  Step 11: Set START->PREV=NEW_NODE
  Step 12: Stop

Circular Doubly Linked List-Deletion

- Steps/Algorithm [Deleting at beginning]:

  Step 1: If START=NULL
  Write UNDERFLOW
  GO TO Step 9
  [End of IF]
  Step 2: Set PTR=START
  Step 3: Repeat Step 4 while PTR->NEXT!=START
  Step 4: Set PTR=PTR->NEXT
  [loop ends]
  Step 5: Set PTR->NEXT=START->NEXT
  Step 6: Set START->NEXT->PREV=PTR
  Step 7: FREE START
  Step 8: Set START=PTR->NEXT
  Step 9: Stop

Circular Doubly Linked List-Deletion

- Steps/Algorithm [Deleting at end]:

Step 1: If START=NULL
Write UNDERFLOW
GO TO Step 8
[End of IF]
Step 2: Set PTR=START
Step 3: Repeat Step 4 while PTR->NEXT!=START
Step 4: Set PTR=PTR->NEXT
[loop ends]
Step 5: Set PTR->PREV->NEXT=START
Step 6: Set START->PREV=PTR->PREV
Step 7: FREE PTR
Step 8: Stop

Circular Doubly Linked List: Assignments

1. Write a program to create a circular doubly linked list.
2. Write a program to display a circular doubly linked list.
3. Write a program to insert a node at the beginning of a circular doubly linked list.
4. Write a program to insert a node at the end of a circular doubly linked list.
5. Write a program to delete a node from the beginning of a circular linked list.
6. Write a program to delete a node from the end of a circular linked list.
7. Write a program to delete a given node of a circular doubly linked list.
8. Write a program to delete the entire circular doubly linked list.

Multiple-Choice Questions (MCQ)

1. A linked list is a
(a) Random access structure (b) Sequential access structure (c) Both (d) None of these
2. An array is a
(a) Random access structure (b) Sequential access structure (c) Both (d) None of these
3. Linked list is used to implement data structures like
(a) Stacks (b) Queues (c) Trees (d) All of these
4. Which type of linked list contains a pointer to the next as well as the previous node in the sequence?
(a) Singly linked list (b) Circular linked list (c) Doubly linked list (d) All of these
5. Which type of linked list does not store NULL in next field?
(a) Singly linked list (b) Circular linked list (c) Doubly linked list (d) All of these
6. Which type of linked list stores the address of the header node in the next field of the last node?
(a) Singly linked list (b) Circular linked list (c) Doubly linked list (d) Circular header linked list
7. Which type of linked list can have four pointers per node?
(a) Circular doubly linked list (b) Multi-linked list (c) Header linked list (d) Doubly linked list

Multiple-Choice Questions (MCQ)

Answers

1. (b) 2. (c) 3. (d) 4. (c) 5. (b) 6. (d) 7. (b)