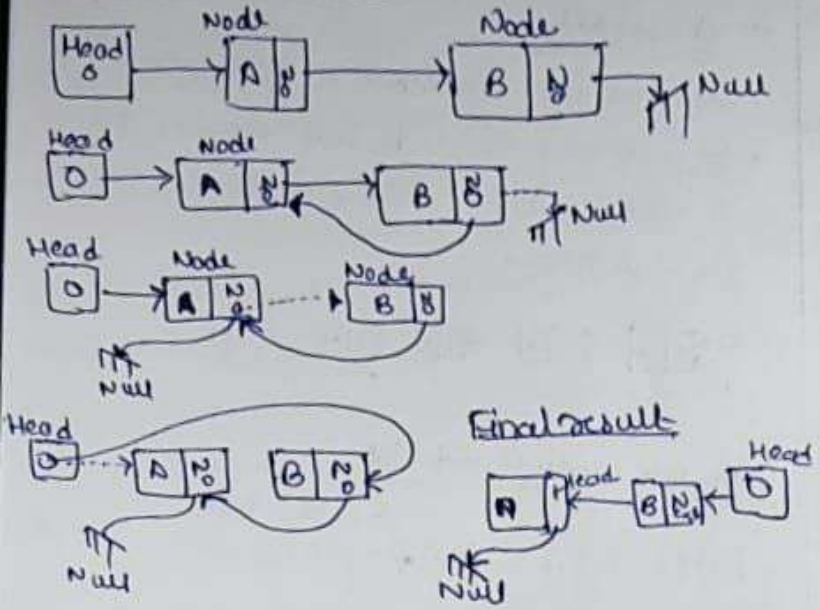
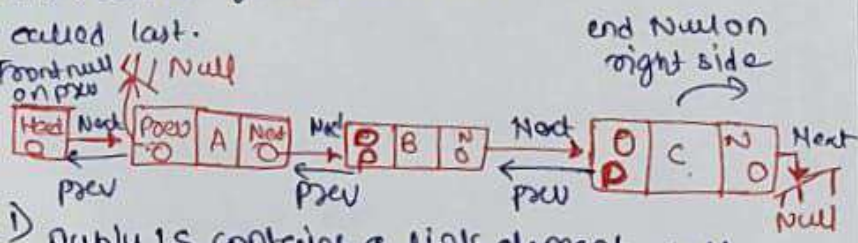
③ Reverse Operation



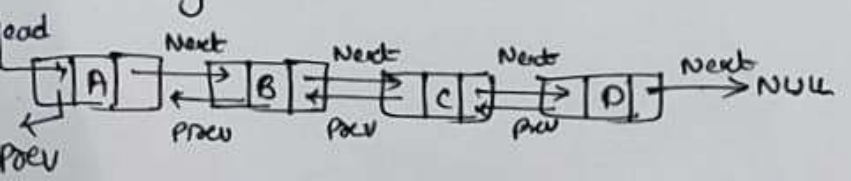
II DOUBLY LINKED LIST

- the list one can move / traverse in both the direction forward & backward.
- Link – each link of a linked list can store a element called data.
- Next – each link of LS contains a link to the next link called next
- Prev – each link of LS contains a link to the previous link called prev.
- LinkedList – LS contains the connection link to the first link called first & to the last called last link called last.



1) Doubly LS contains a link element call first and last
2) Each link carries a data field (s) & two link fields called next & prev
3) each link is linked with its previous link using next link → prevlo

Doubly linked list



Basic Operations

- Insertion – Adds an element at the biginning of the list
- Deletion – Delete an element at the Beginning of list
- Insert last – Adds an element at the end of list
- Delete last – Deletes an element at end of the list
- Insert After – Adds an element after an item of the list
- Delete – Delete an element from list using key
- Display forward – Display complete list in forward manner.
- Display backward – Display complete list in a backward manner.

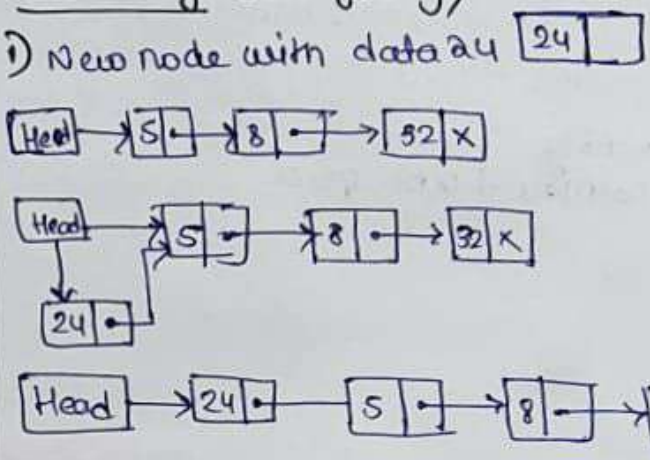
ALGORITHMS IN LINKED LIST

Traverse

1) Step 1: [INITIALISE] SET PTR = HEAD
Step 2: Repeat Step 3 and Step 4 while PTR != NULL
Step 3: Apply process to PTR → DATA
Step 4: SET PTR = PTR → NEXT
[END OF LOOP]
Step 5: EXIT

Inserting (at bagining)

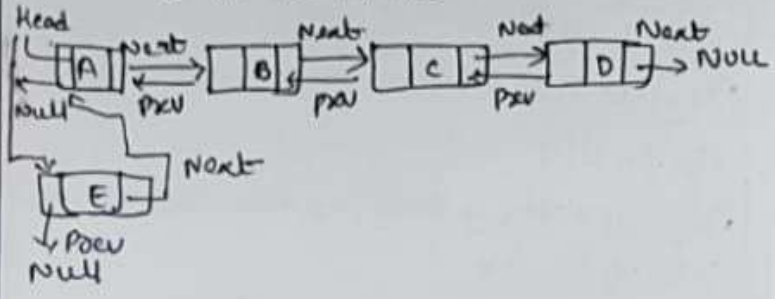
1) New node with data 24



Insertion at DLL - Node can be added in four ways

1) Front 2) after given node 3) at end 4) Before a given node.

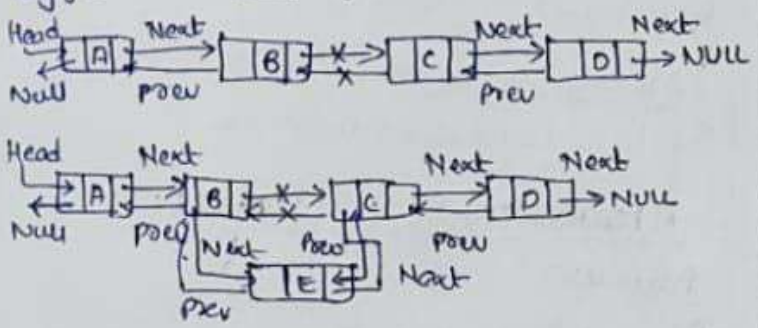
1) Add new node at front - Always add before head node eg:- 1→0→1→5 we add an item 5 5→1→0→1→5



October 5

Add node after given node

eg:- add [E] after B



Working of DLL

The left pointer points to the node which is before the current node & the right pointer points to the node after current node. + Backward Traversing.

Applications

- Traverse both direction . Insertion easy
- Deletion easy . Tree Datastructure . Undo/Redo operations → web page navigation
→ used in games like a deck of cards.

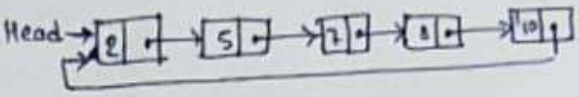
Disadvantage

→ Two pointers + extra space

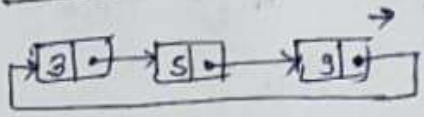
circular.

- All nodes are connected in a form of circle.
- 1st node and last node are connected to each other which forms a circle.

No Null at end.



1) circular singly linked list



→ last node contains a pointer to 1st node.

- No Null . Traversing is done till we reach same position.

2) circular Doubly linked list (Doubly linked list + circular)