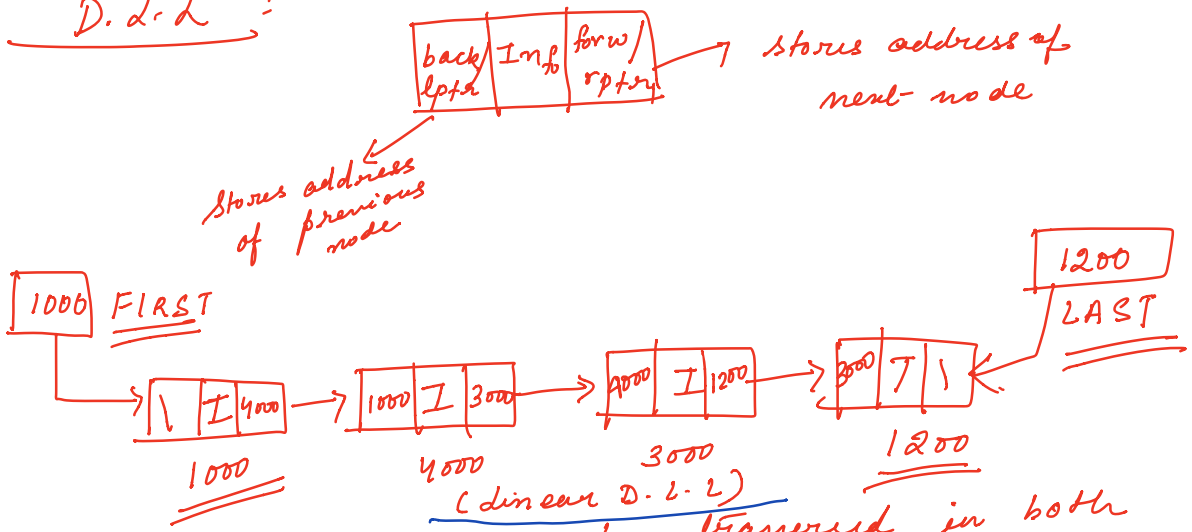
Lec-08 03/05/21

→ Doubly d-list / 2-way list
→ Applications of d-list
  → Poly. addition
  → Poly. multiplications
  → Linked dictionary
→ Concept of garbage collection / compaction

D.L.L :



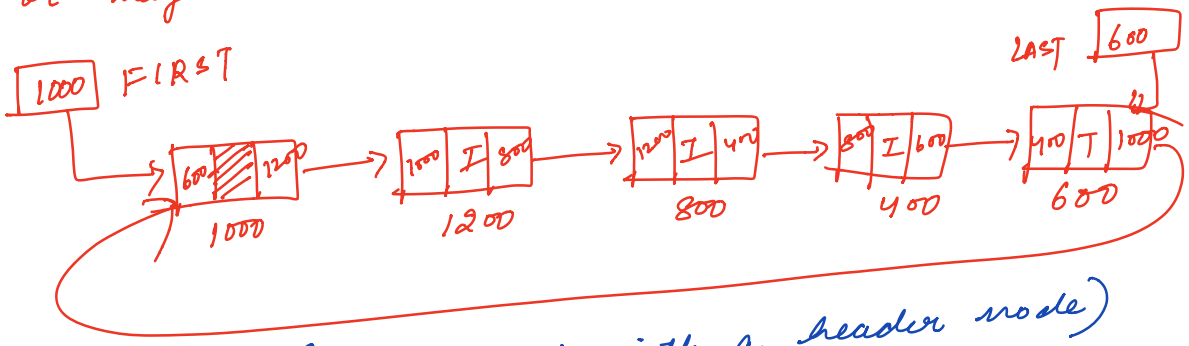
The 2-way list can be traversed in both directions → (i) forward direction (begin to end)
(ii) backward direction (end to begin)
(linear D.L.L)

defn: A 2-way list is a linear collection of data elements called Nodes where each node N is divided into 3 parts:

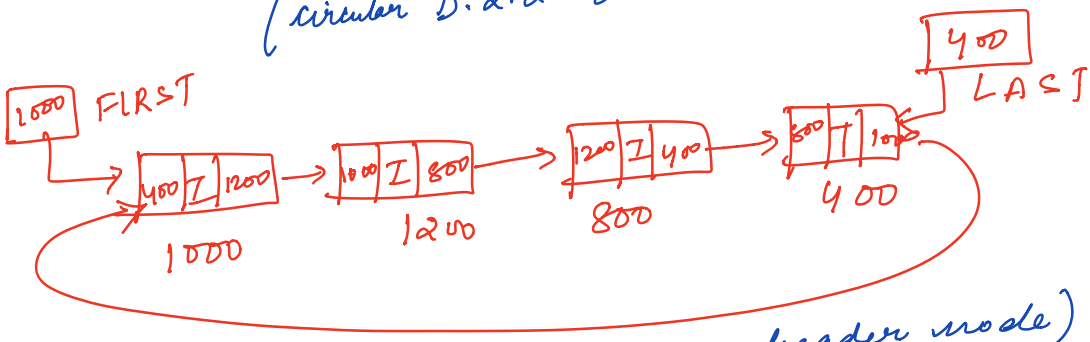
i) An information field INFO contains user informations.
ii) A pointer field FORW (rptr) which contains the address of next node in the d-list.
iii) A pointer field BACK (lptr) which contains the address of previous node in the d-list.

→ The list contains two list pointer variables FIRST and LAST which stores the address of first node and last node of D.L.L. resp.

* D.L.L. may be either linear, circular may or may not contain a header node.



(circular D.L.L with a header node)



(circular D.L.L without a header node)

1) DLL- insert :

This algo. inserts a data item X to the left of a node whose address is M in a D.L.L. L & R represent the left- most and right most node of D.L.L

1) if avail = NULL, write "overflow" and return.

2) new = avail
   avail = link (avail)

3) set info (new) = X

4) if L = R = NULL then // Insertion to an empty list

LPTR(new) = RPTR(new) = NULL

set L = R = New

return

5) if M = L then

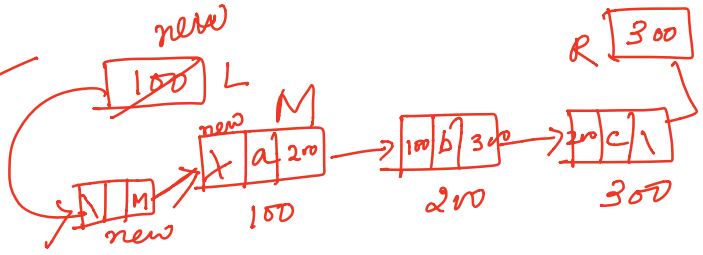
LPTR(new) = NULL ✓

RPTR(new) = M

LPTR(M) = new

L = new

return



↳ // Insert a node as left-most node

6) // M is in middle. so insert new node left of M //

LPTR(new) = LPTR(M) ✓

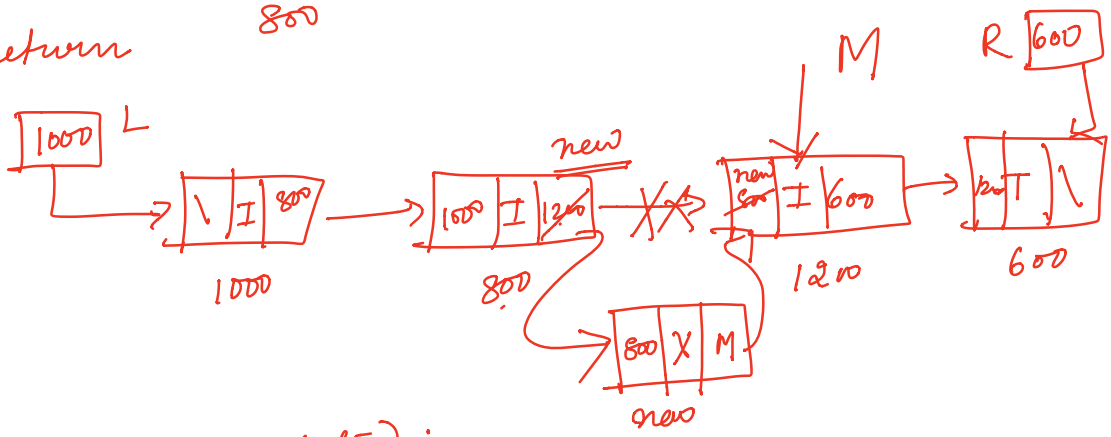
RPTR(new) = M

LPTR(M) = new

RPTR(LPTR(new)) = new

800

return



2 D.L.L (delete):

This algo. deletes a node with an address Loc from D.L.L with L and R as the pointers storing the address of left-most and right-most nodes resp.

Step 1) if L = R = NULL, write "underflow"; return.

2) if L = R ≠ NULL // single node before deletion

then set L = R = NULL

else if DOC = L // delete the left-most node

then L = RPTR(L)

LPTR(L) = NULL

else if DOC = R // delete the right-most node

then R = LPTR(R)

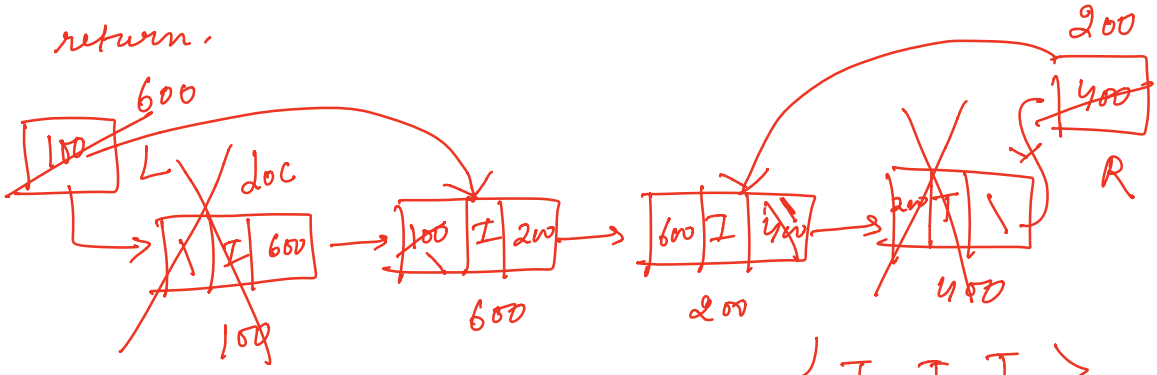
RPTR(R) = NULL

else // delete from middle

RPTR(LPTR(DOC)) = RPTR(DOC)

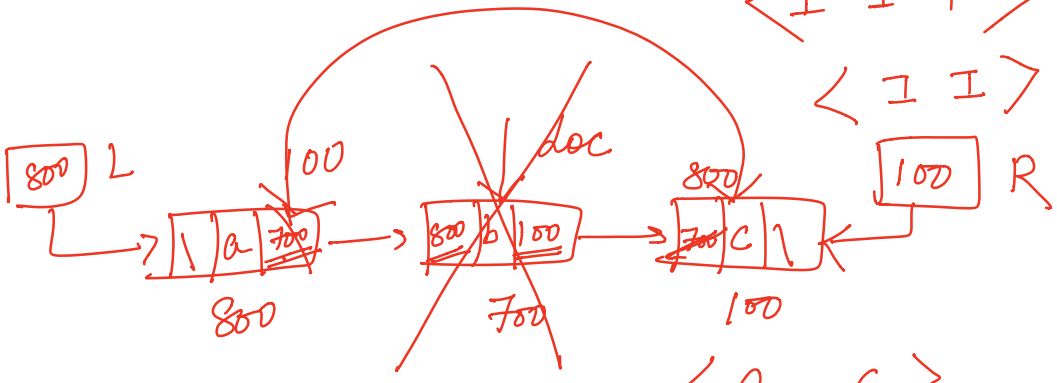
LPTR(RPTR(DOC)) = LPTR(DOC)

3. return.



<I I T>

<I I>



<a, c>