Lec. 06 28/04/21

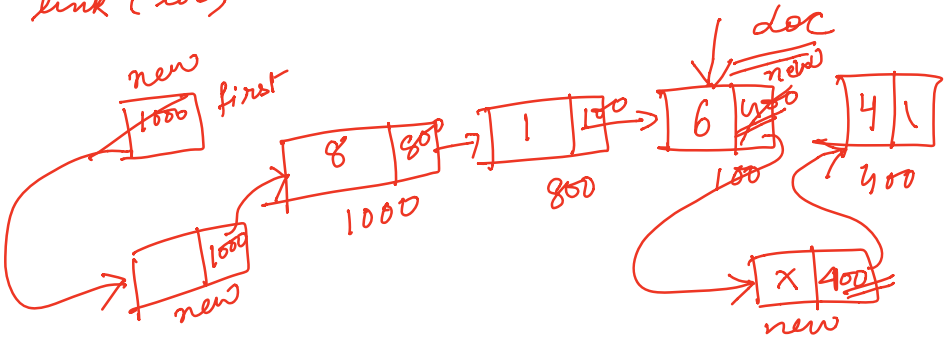
→ Ins - loc - after
→ Ins - loc - before
→ del - first
→ del - last
→ del - loc

→ Ins - loc - after :

This algo. inserts a node (whose info field contains X) after a node whose address is given as loc.

If loc = NULL, then the node is created as first node.

1. if avail = NULL, write "overflow" and return
2. new = avail, avail = link (avail)
3. info (new) = X
4. if loc = NULL, then link (new) = FIRST
   set FIRST = new
   else link (new) = link (loc)
   link (loc) = new .
5. Exit.



Ins - loc - before :

This algo. is used to insert a node with information containing X before a node with address loc.

If loc = NULL, then the node is inserted as first node

1. if avail = NULL, write "overflow" and return .
2. new = avail, avail = link (avail)

3. info (new) = X

4. if loc = NULL, or loc = FIRST then
   link (new) = FIRST
   set (FIRST) = new)

return

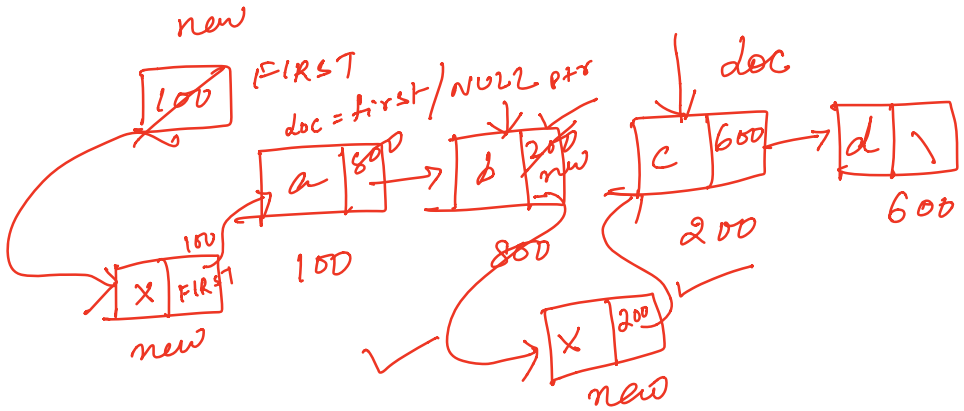
5. set ptr = FIRST

6. repeat while link (ptr) ≠ loc
   ptr = link (ptr)

7. link (new) = link (ptr)
   link (ptr) = new)

ptr	loc
~~100~~ 800	200

8. Exit



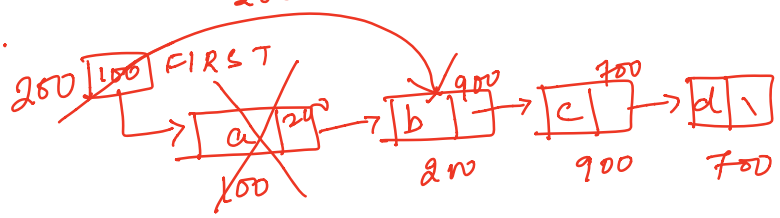
deletion - first :

This algo. deletes the first node from a list.

1. if FIRST = NULL, then write "underflow", return.

2. FIRST = link (FIRST)

Exit.



del - last :

This algo. deletes a node from the last posn. of linked list.

1) if FIRST = NULL, write "underflow", return -

2) if link(FIRST) = NULL then FIRST = NULL // L.list has only one node

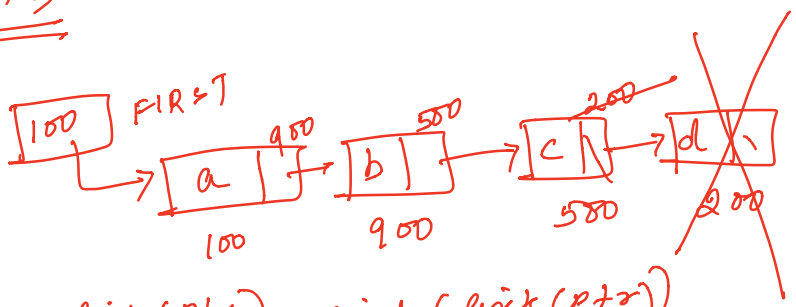
else

3) Set PTR = FIRST
repeat (step 4) while (link(link(ptr))) ≠ NULL

4) ptr = link(ptr)

5) link(ptr) = NULL

6) Exit



ptr	link(ptr)	link(link(ptr))
~~100~~	~~900~~	~~500~~
~~900~~	~~500~~	~~200~~
500	200	NULL

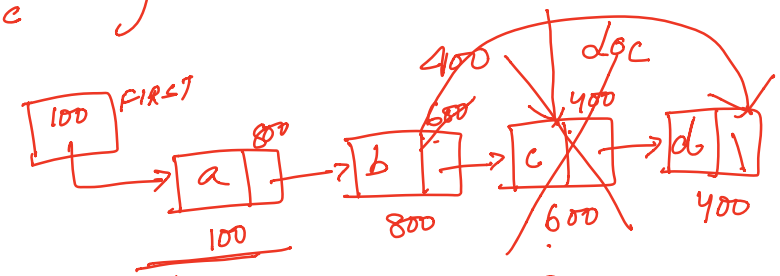
Del - loc :-

This algo. deletes a node with address loc from the L.list whose first node address is stored in FIRST pointer.

1. if FIRST = NULL, write "underflow", return.
2. set PTR = FIRST
3. repeat step 4 and 5 while (PTR ≠ loc and link(ptr) ≠ NULL)
4. PRED = PTR
5. PTR = link(PTR)

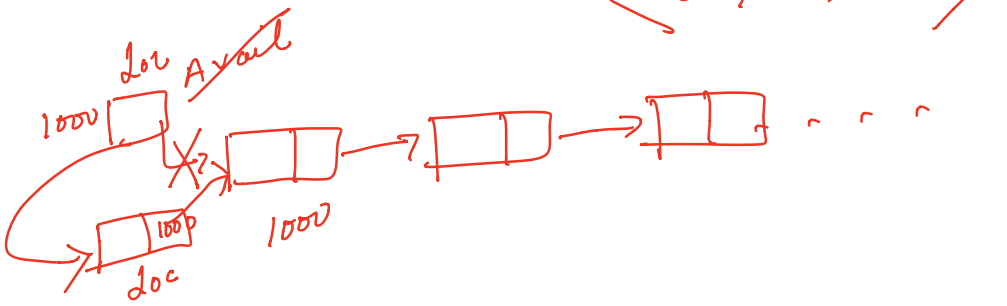
[T & T = T]

6. if PTR ≠ loc, print "node not-found", return.
7. if PTR = FIRST then FIRST = link(FIRST)
   else link(PRED) = link(loc)
8. link(loc) = avail
   avail = loc
   } return node to avail list
9. Exit.



Pred	Ptr	loc	link(ptr)
~~100~~ 800	~~100~~ ~~800~~ 600	600	~~800~~ 600

< a, b, d >



* Reverse of a list (use stack DS) to be covered later....