Lecture : 18

Deletion in a Linked List

First we discuss the algorithm that finds the location LOC of the node N which contains ITEM and location LOCP of the preceding node N. If ITEM does not appear in the list, then the procedure sets LOC = NULL & if item appear in the first node, then it sets LOCP = NULL

FIND ( )

1. If start = NULL then: // If the list is empty
   Set LOC = NULL and LOCP = NULL & Return
   (If this becomes false)
2. If INFO[start] = ITEM, then
   Set LOC = start
   and LOCP = NULL
   & Return
   } when ITEM appears in the first node.
   (If this becomes false)
3. Set SAVE = start and Ptr = LINK[start] // ~~update the two pointers~~ ie we initialize them.
4. Repeat step 5 & 6 while ptr ≠ NULL
5. If INFO[Ptr] = ITEM
   Set LOC = PTR
   and LOCP = SAVE and return
6. Else set SAVE = Ptr
   and PTR = LINK[Ptr] // update the pointers.
7. Set LOC = NULL // search is unsuccessful.
8. Return

This algorithm deletes from a linked list the node N which contains the given item of information.

~~1. Call~~ Delete ( )

1. Call FIND ( ) algorithm for the location.

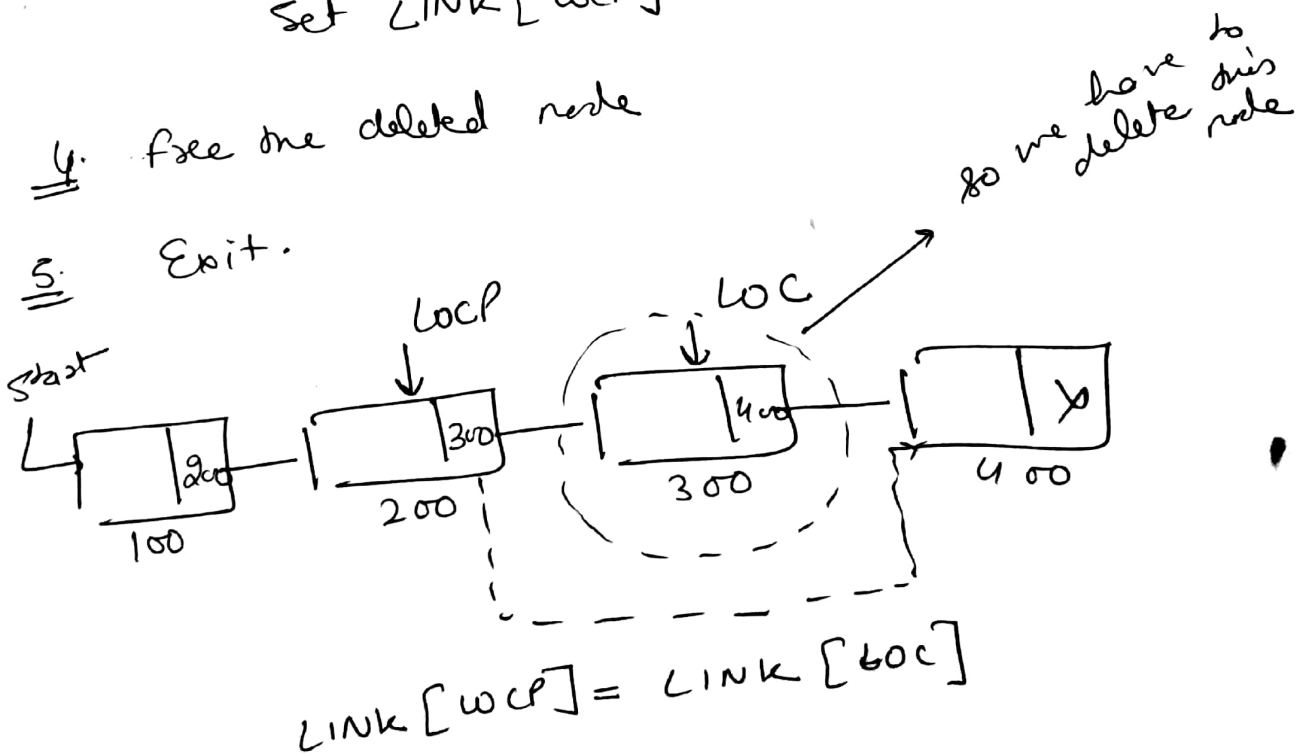
2. If LOC = NULL then
   Write ITEM not in the list & Exit.

if this become false

3. If LOCP = NULL then || delete the first node
   Set start = LINK [start]
   Else
   Set LINK [LOCP] = LINK [LOC]

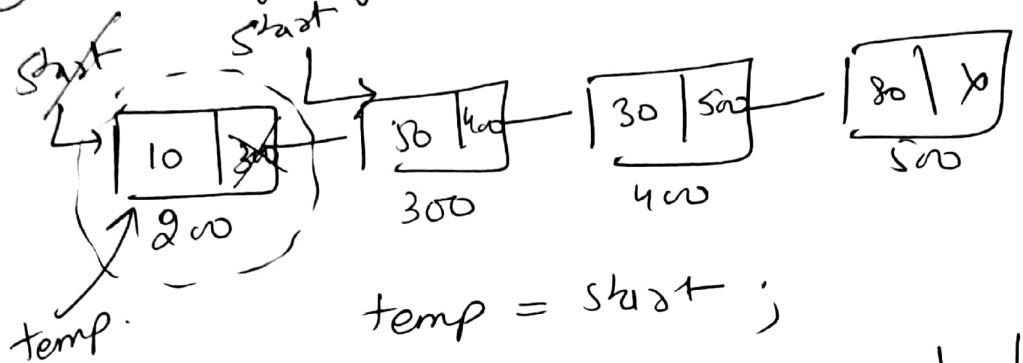
4. free the deleted node

5. Exit.



Showing the pictorial representation of three case for deletion

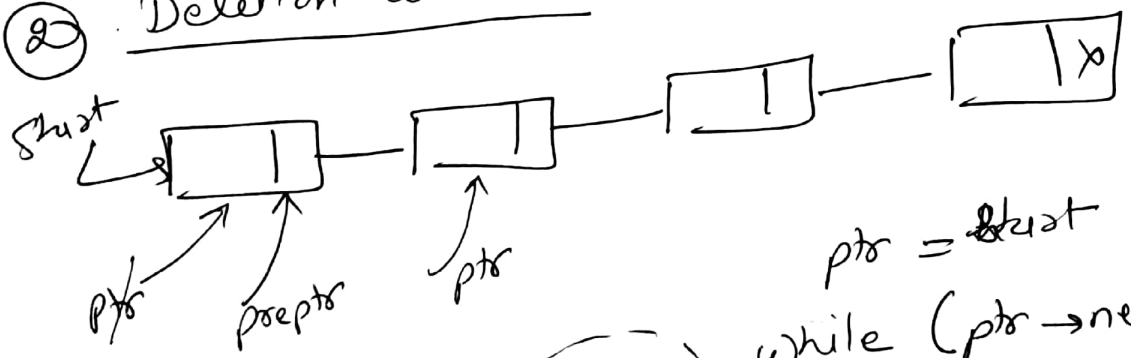
① At Beginning of the list

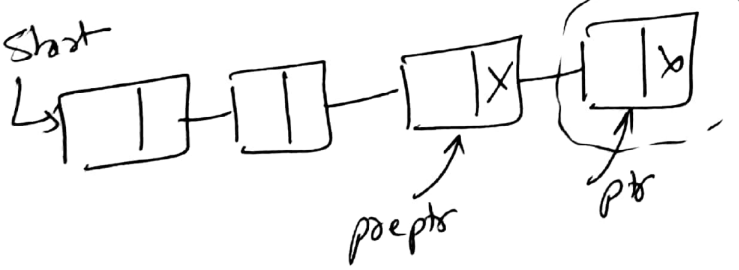


temp.

```
temp = start ;
start = start → next || update the start
temp → next = NULL
free (temp);
```

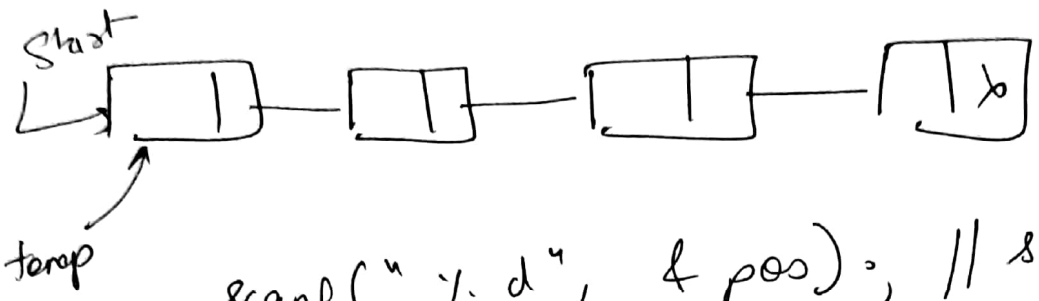
② Deletion at End






```
ptr = start
while (ptr → next != NULL)
{
  preptr = ptr || initialize a new pointer
  ptr = ptr → next || update ptr
}
preptr → next = NULL
free (ptr);
```

At a specific Position



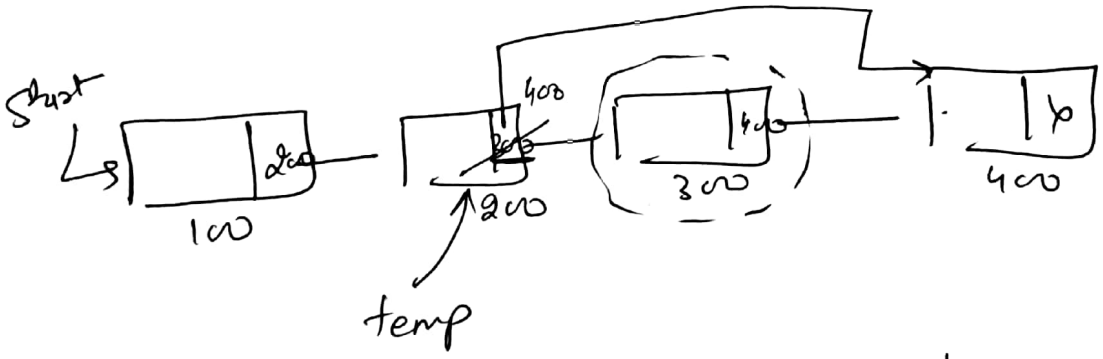

```
scanf("%d", &pos); // say we read 2

temp = start;

for( i=0 ; i<pos-1 ; i++)

{
  temp = temp→next; // update the pointer.
}
```

temp → next = temp → next → next



temp → next = temp → next → next

300

400.

But Better is to use two pointers in case of single linked list when you perform deletion. Because in above case we are not able to free the deleted node. It is disconnected But still hold the space.