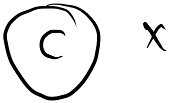
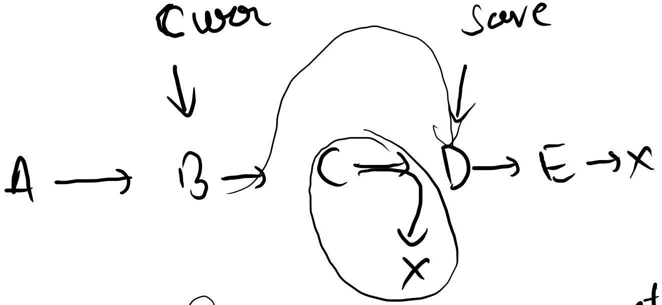
Delete a <u>node</u>.



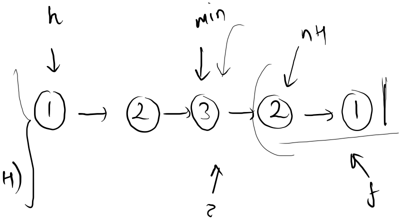


```
save = curr.next.next
curr.next.next = null.

curr.next = save
```

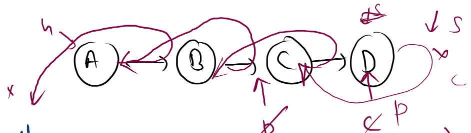
Palindrome

1. mid
2. nH
3. revers e(nH)
4. Compare (H, nH)



mid = mid_node(h)

nH = mid.next

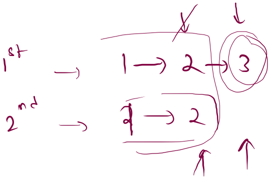


p = null

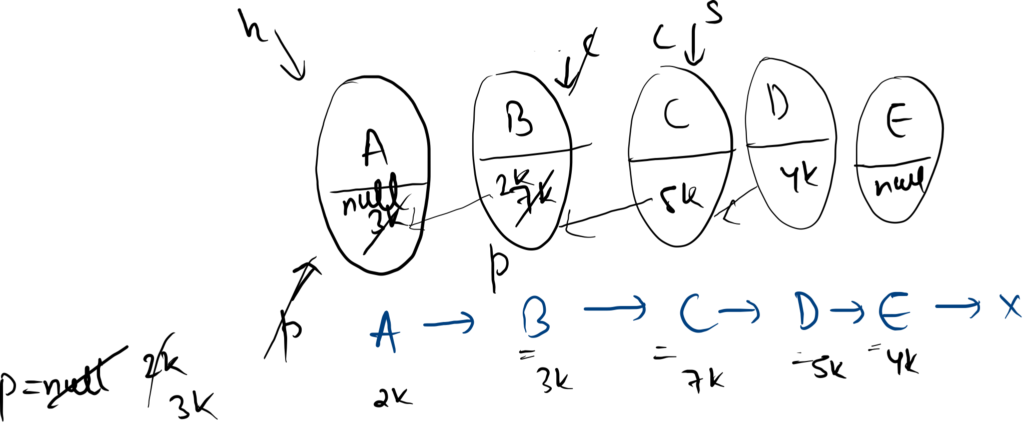
```
~~p~~
while ( c != null )
{
    s = c.next
    c.next = p    null
    p = c
    c = s
}
```

(P)

D → C → B → A



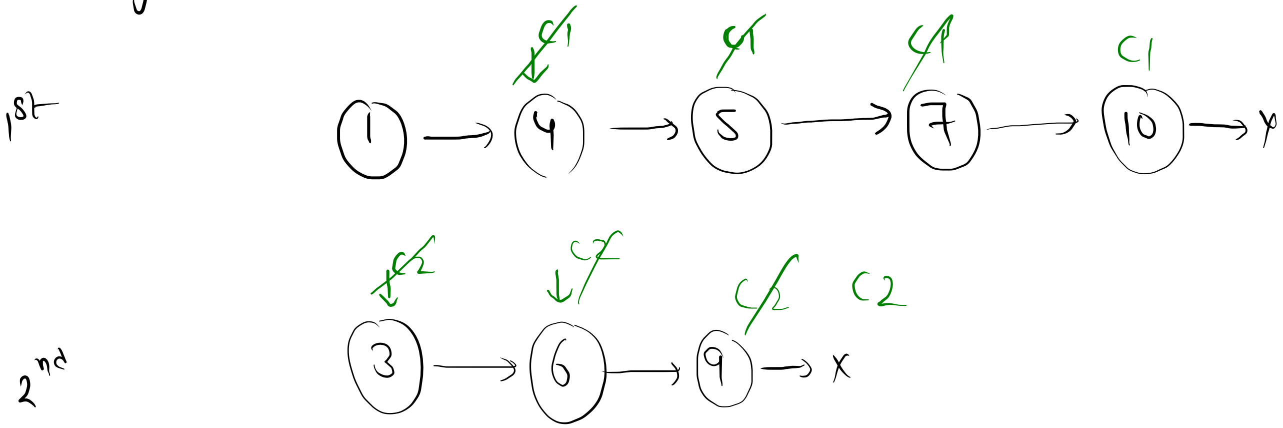
1 2 3 2 1

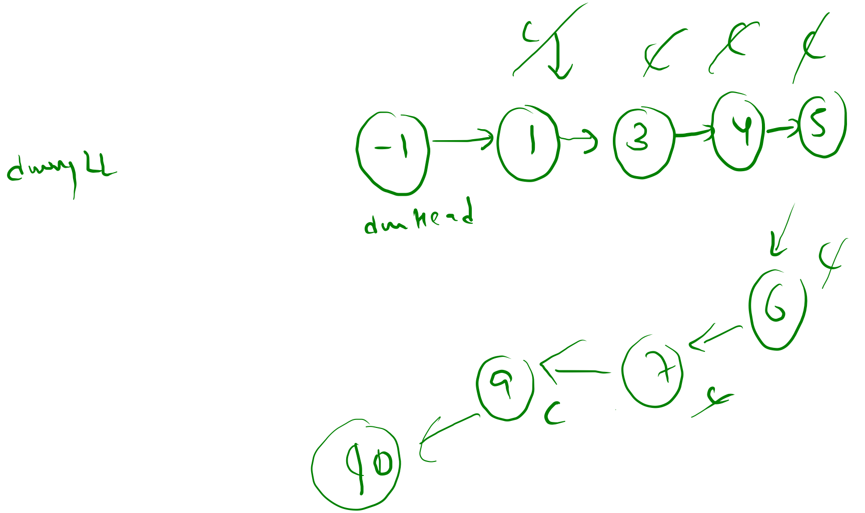


```
c = h

(c != null)
{
    s = c.next
    c.next = p
    p = c
    c = s
}
```

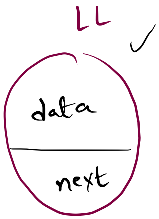
Merge two sorted LL.



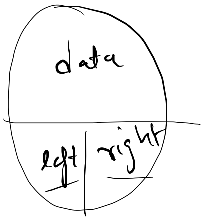


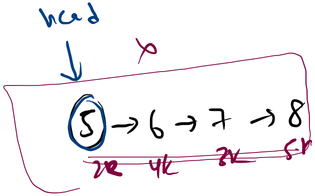
```
(c1 != null && c2 != null)
    if (c1.data < c2.data)
        c.next = c1
        c1 = c1.next
    else
        c.next = c2
        c2 = c2.next.
    c = c.next.
```

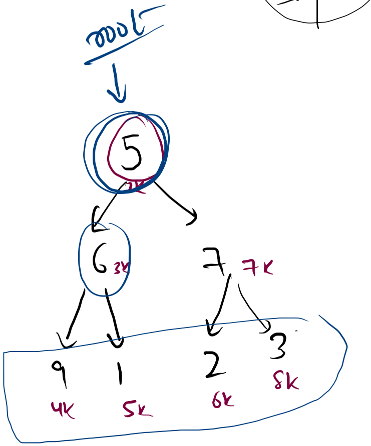
Binary Tree

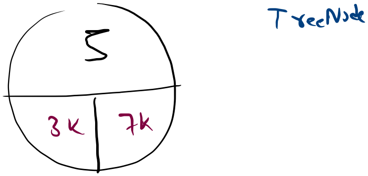


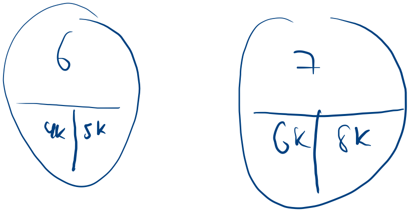
B - Tree



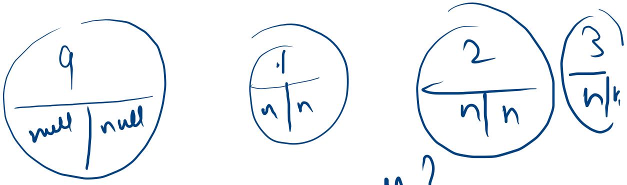








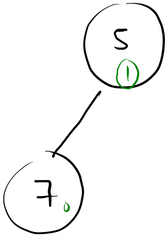
head

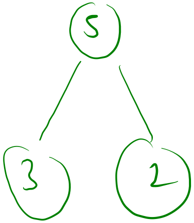


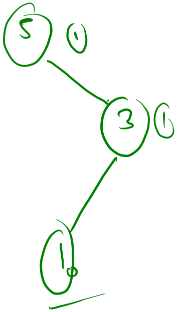
which have left==null && right==null } leaf nodes

Binary Tree

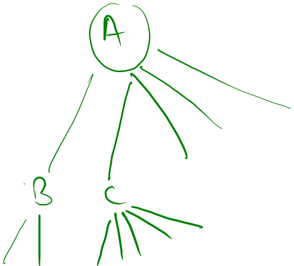
↳ A node can have at max 2 child.



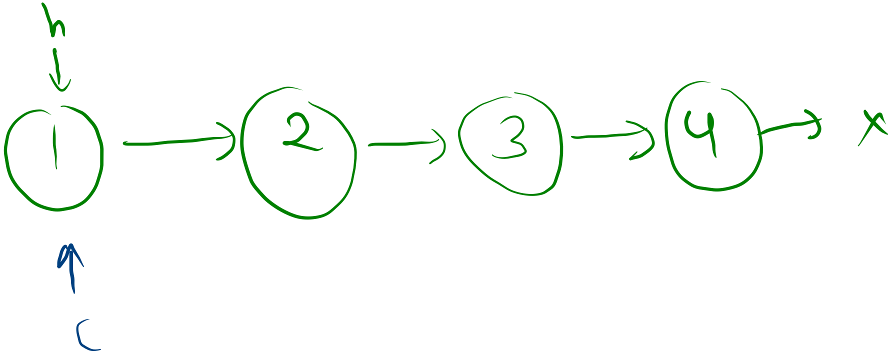




Generic - Tree (n-ary tree).



print. LL

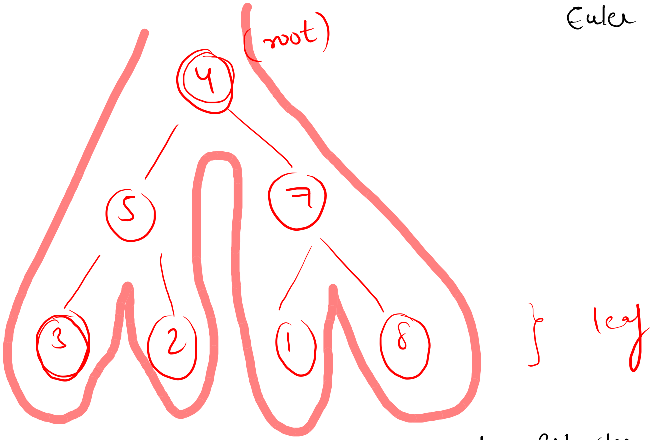


iterative.

```
c != null
{
    sys.o( c.data );
    c = c.next
}
```

print BT. (Recursive Algo).

Euler Path.



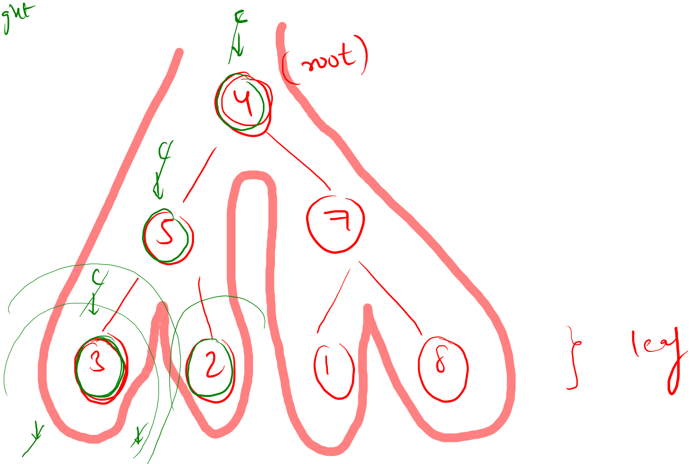
Preorder traversal.
→ Root left Right

Inorder Traversal
left Root Right

Postorder Traversal
Left Right Root

Pre= Root left right

4 5 3 2


(root)
4
5
7
3
2
1
8
leaf