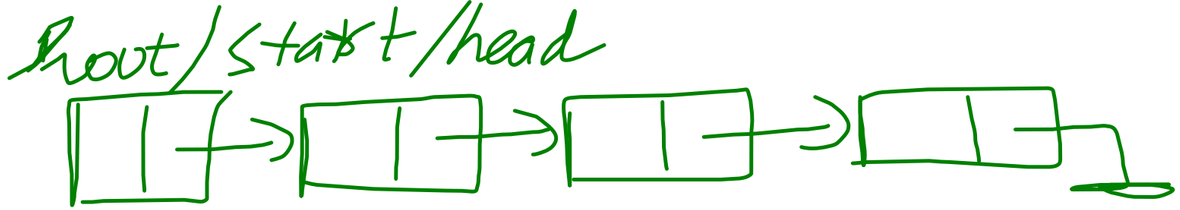
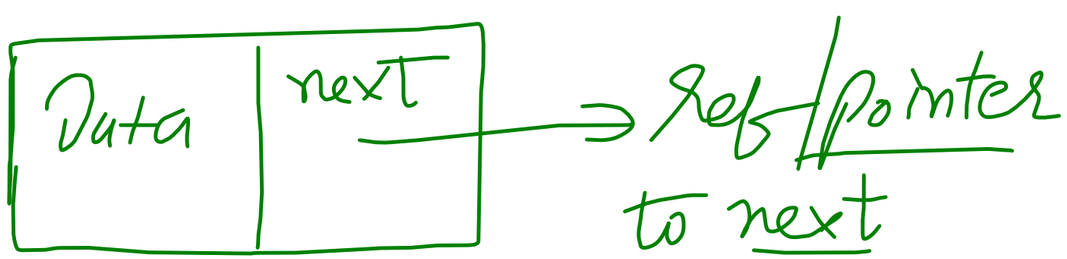
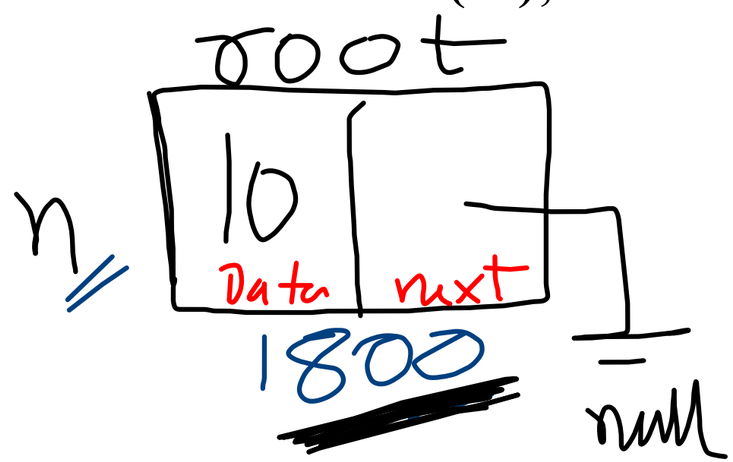
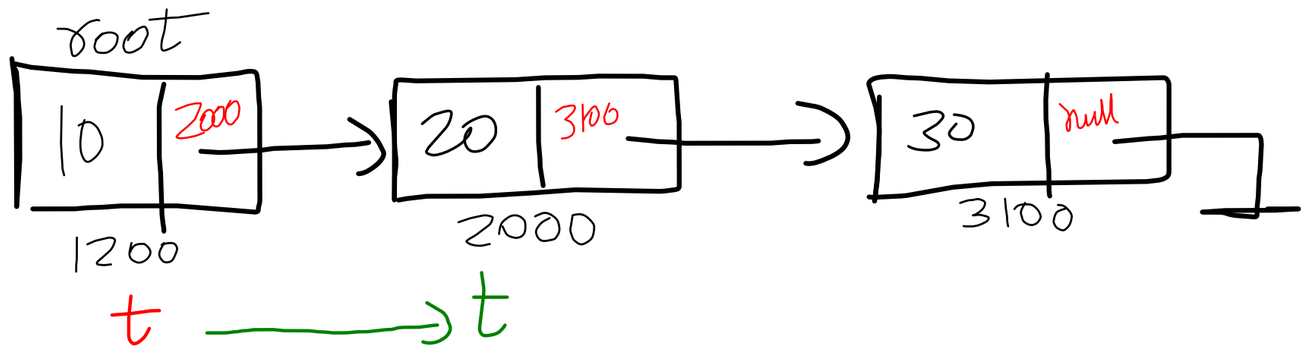


Linked List:linear,dynamic,can operate from any side
Collection of nodes arranged in a sequential manner.



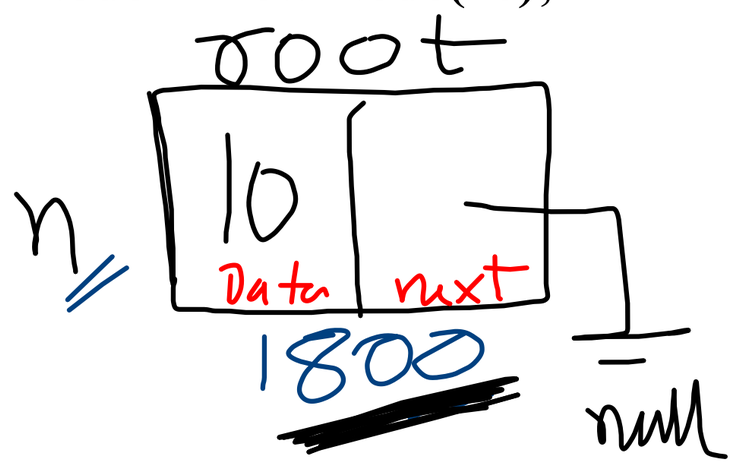
Node n=new Node(10);

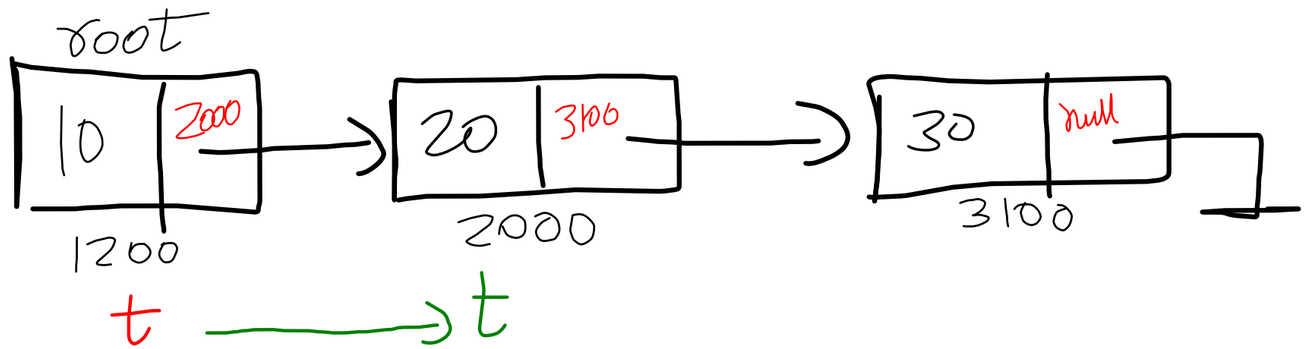




Node t=root; t=1200
t=t.next; t=2000(move to next)

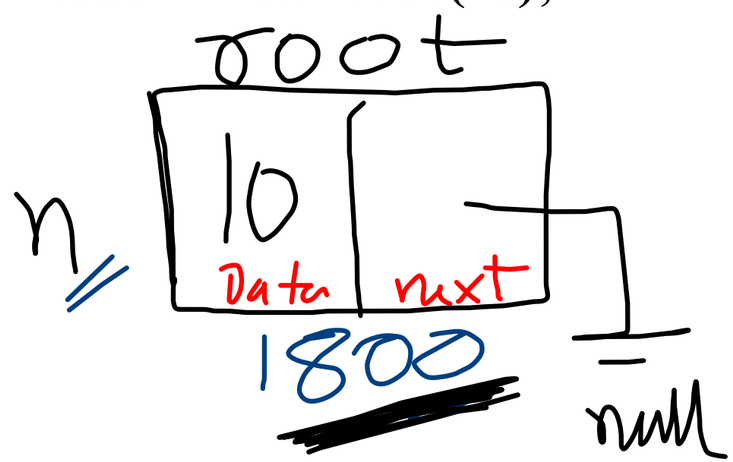
Node n=new Node(10);

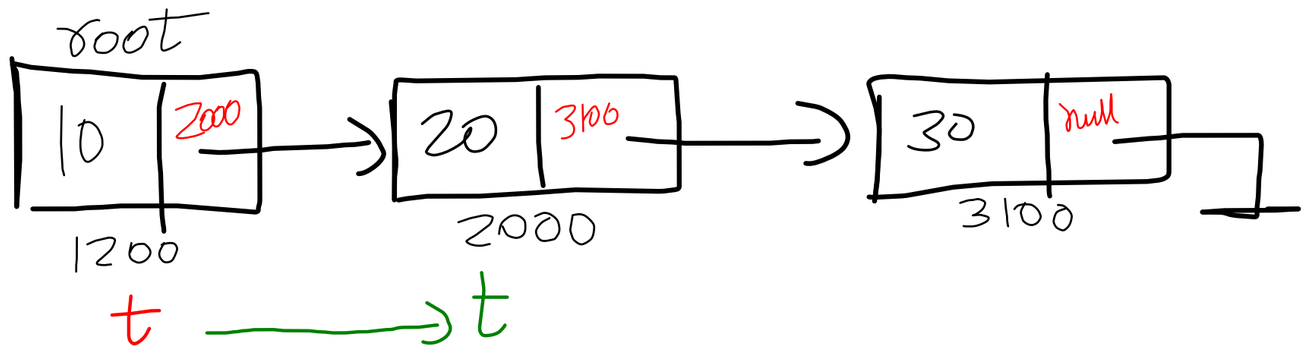




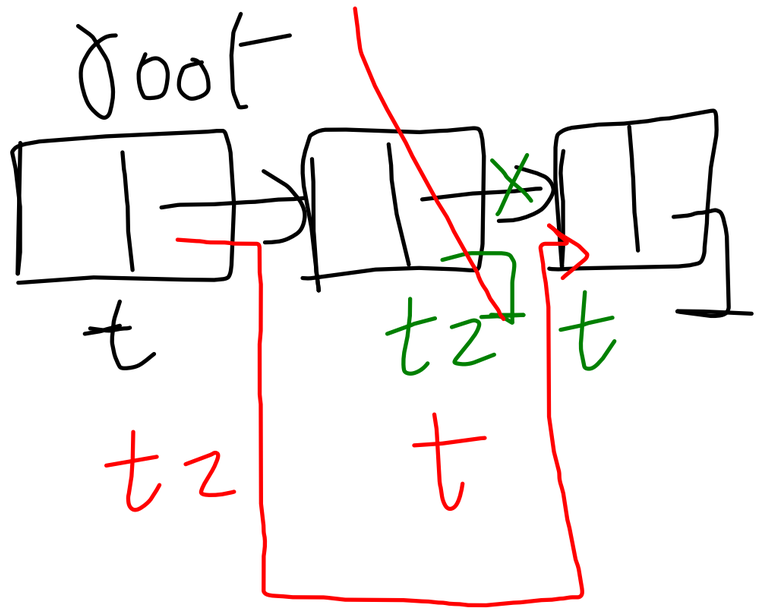
Node t=root; t=1200
t=t.next; t=2000(move to next)

Node n=new Node(10);



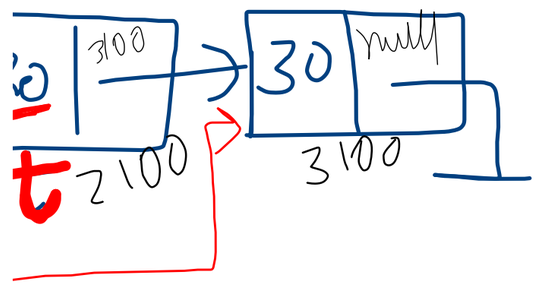


Node t=root; t=1200
t=t.next; t=2000(move to next)


root
t
t2
t2
t
t

```
void delete_element(int key)//key if found need to be deleted
   {
      if (root == null)//no root
         System.out.println("List is empty");
      else {
         Node t, t2;
         t = t2 = root;//1
         while (t != null)//If found, stop.
         {
            if (t.data == key)
               break;
            t2=t;
            t = t.next;
         }
         if (t == null)//not found
            System.out.println("\n" + key + " Not Found");
         else
         {
            System.out.println("\n" + key + " Found");
            //deletion part
            if(t==root)//case 1:delete left
               root=root.next;
            else if(t.next==null)//case 2 : right most: delete
right
               t2.next=null;//deleted
            else //case 3 in-between deletion
               t2.next=t.next;//redirect to next of t
            System.out.println("\n"+t.data+" deleted.");
      }
   }
}
```

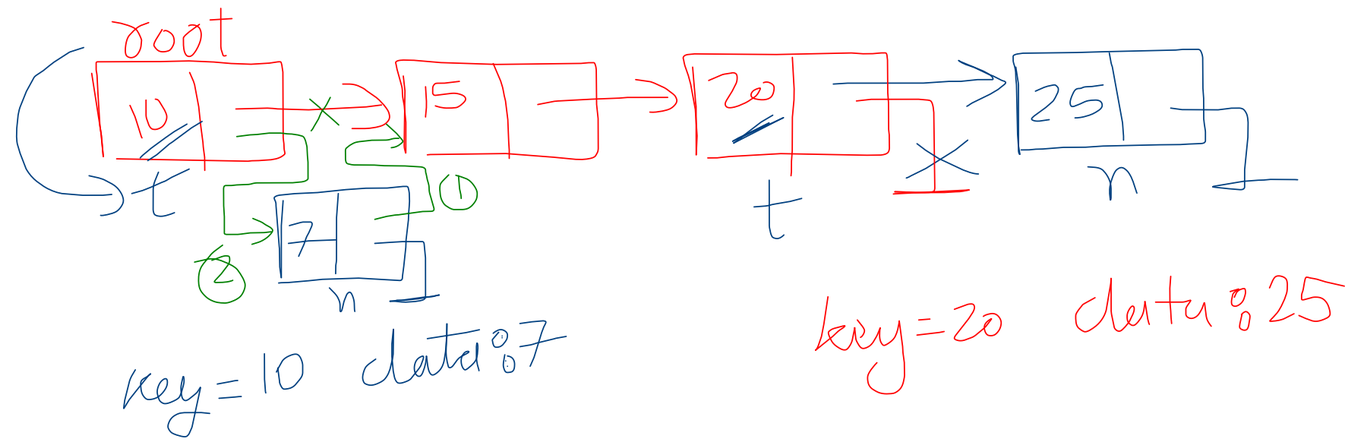
key 20



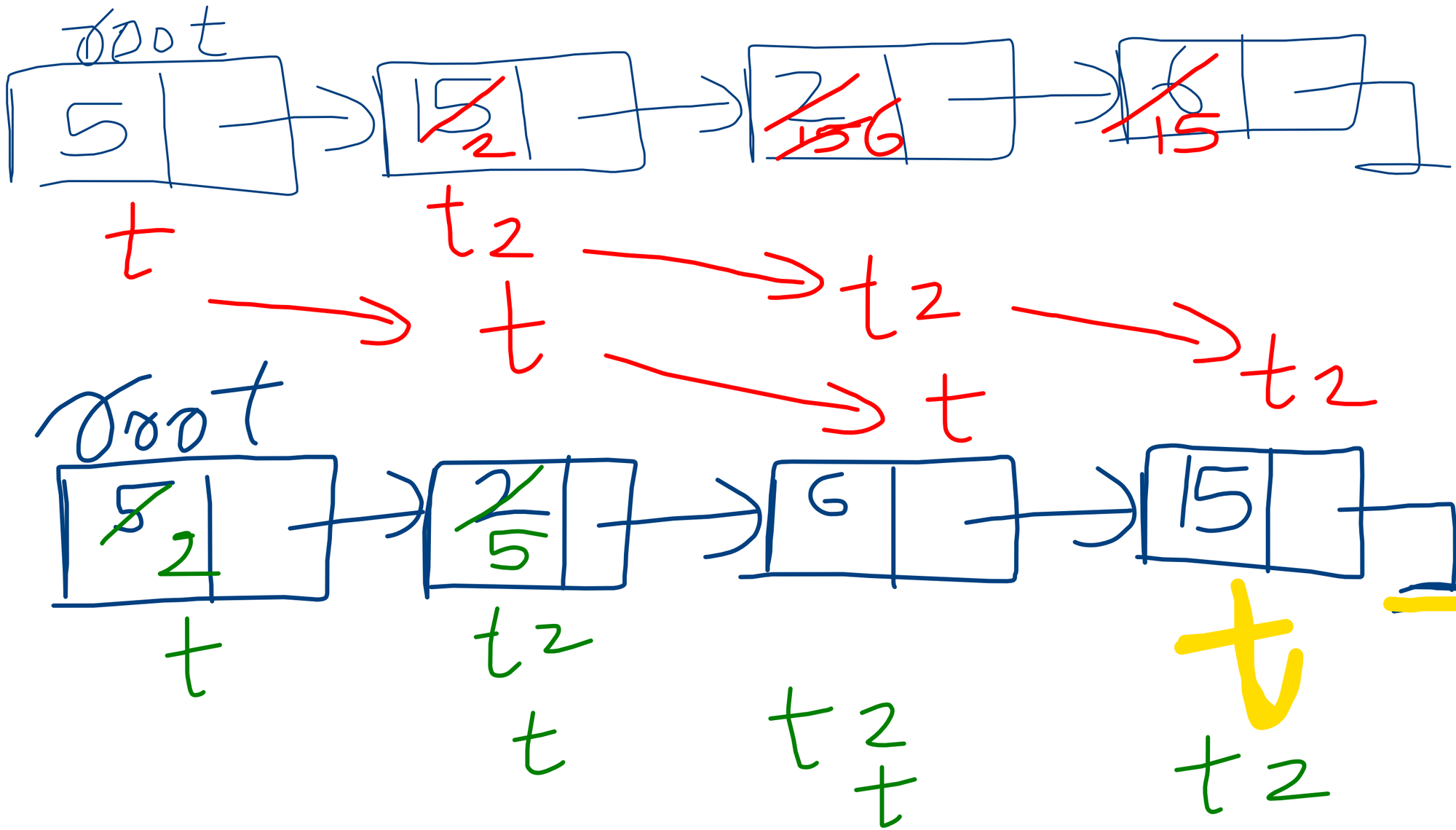
t2.next=t.next;

t2.next=3100;

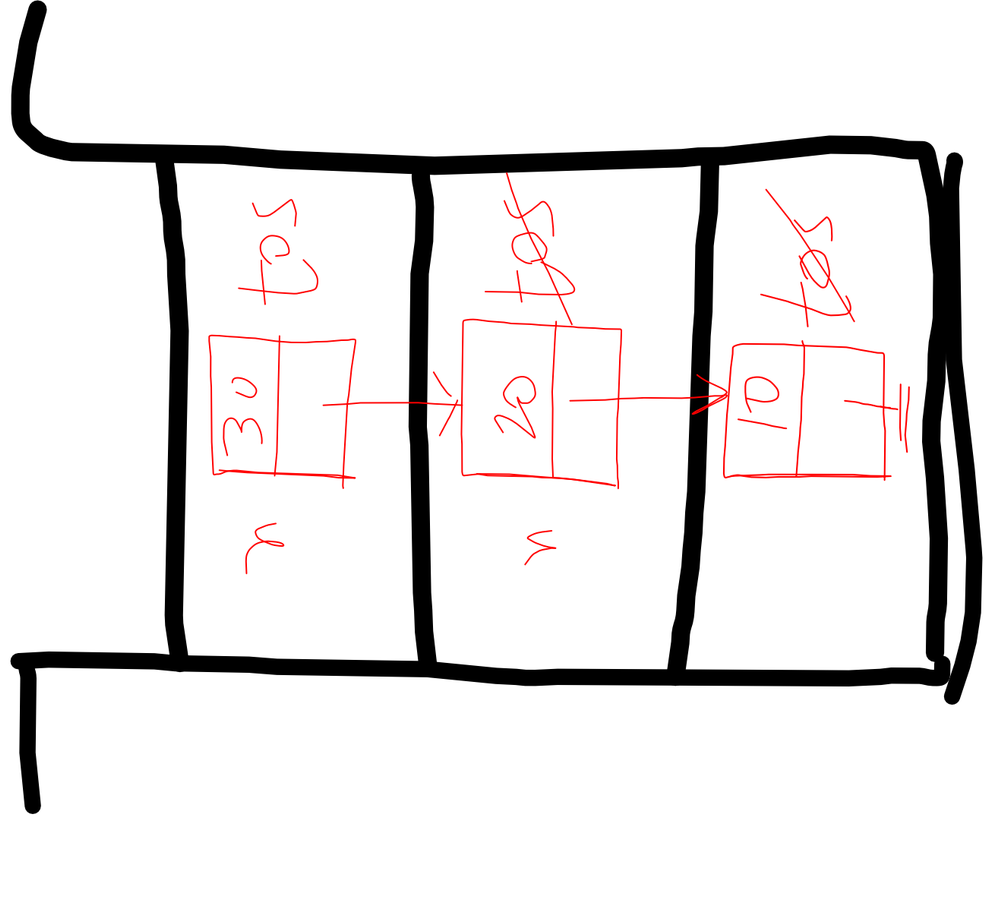
```java
//We'll search for the key element if found.
//We'll create a new node with given data and insert it after that
key node.
void insert_after(int key,int data)
{
    if (root == null)//no root
        System.out.println("List is empty");
    else {
        Node t;
        t = root;//1
        while (t != null)//If found, stop.
        {
            if (t.data == key)
                break;
            t = t.next;
        }
        if (t == null)//not found
            System.out.println("\n" + key + " Not Found");
        else
        {
            System.out.println("\n" + key + " Found");
            Node n=new Node(data);//created new node
            //insertion part
           n.next=t.next;//1 ref to who ever is t.next
            t.next=n;//2 t ref to n

            System.out.println("\n"+t.data+" deleted.");
        }
    }
}
```



Creating in-place/im-place sorting for linked lists.
Permanent changes done to linked list.



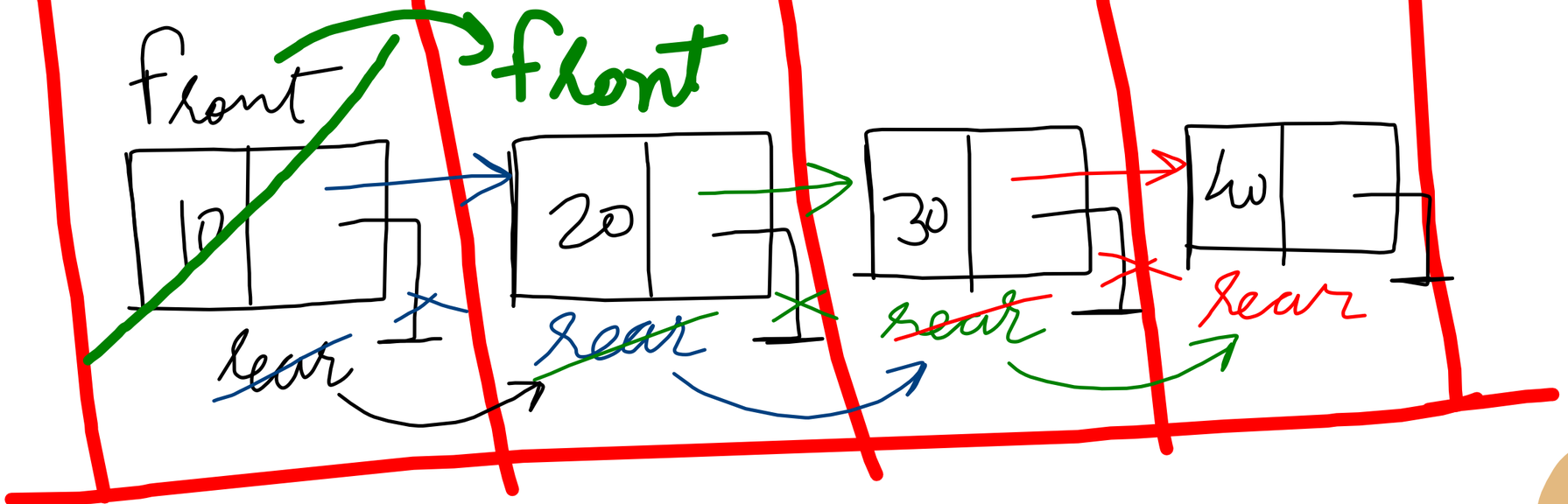
push(10)
push(20)
push(30)



front

Enqueue(10)/(20)/(30)/(40) Dequeue()

rear



L

R

Node: front,rear
Enqueue():insert_right()
Dequeue():delete_left()
print_queue():front to rear

Implement Employee Management System using LinkedList which supports CRUD operations with basic functionality of:

. Register Employee
. Delete Employee
. Search Employee
. Update Employee

All these happen on employee ID.
We have details like:

- Employee name
- Employee ID
- Employee gender
- Employee salary

Implement polynomial addition using linked list.