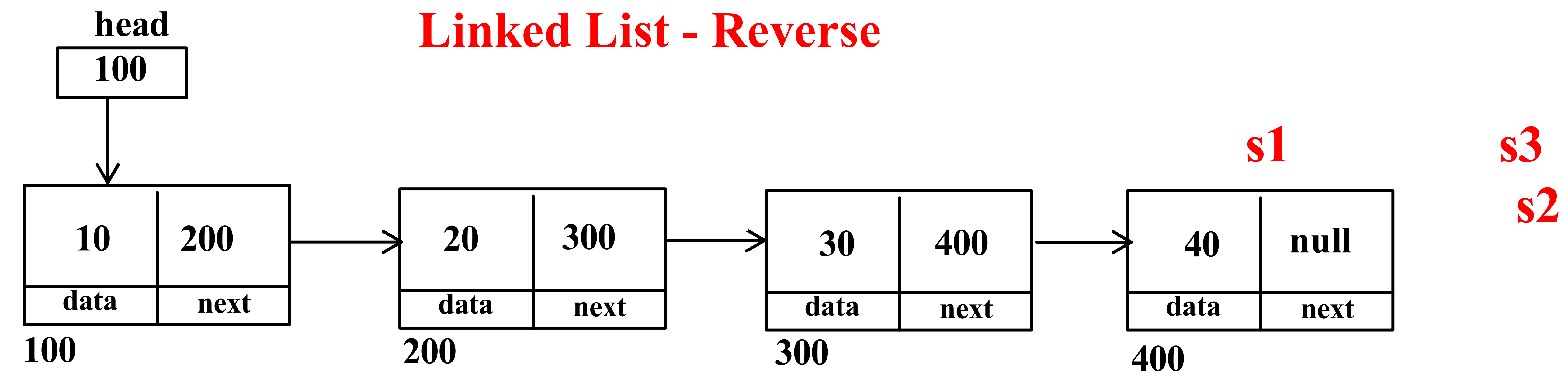
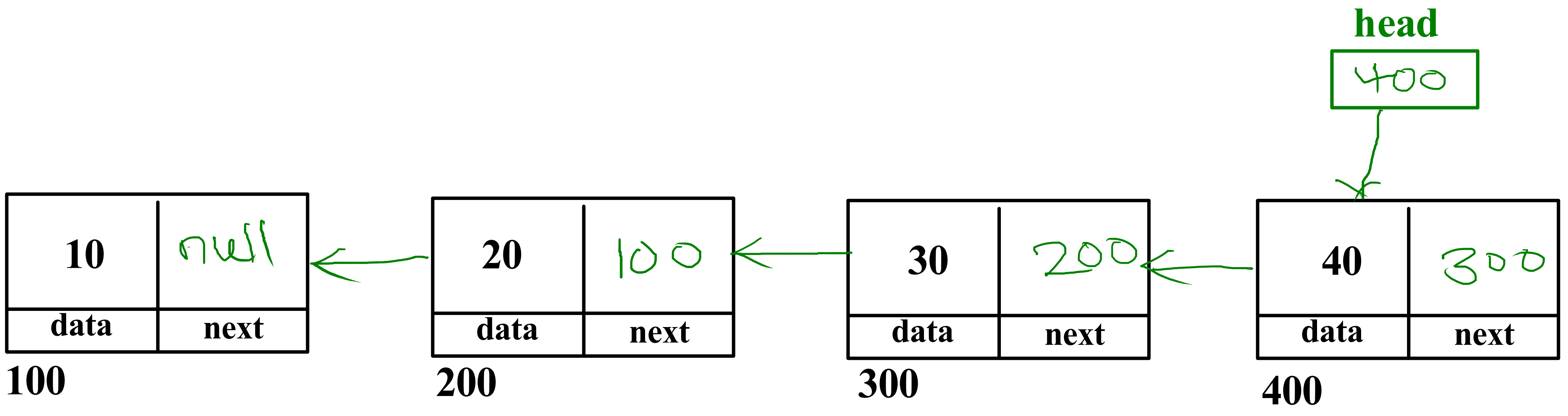
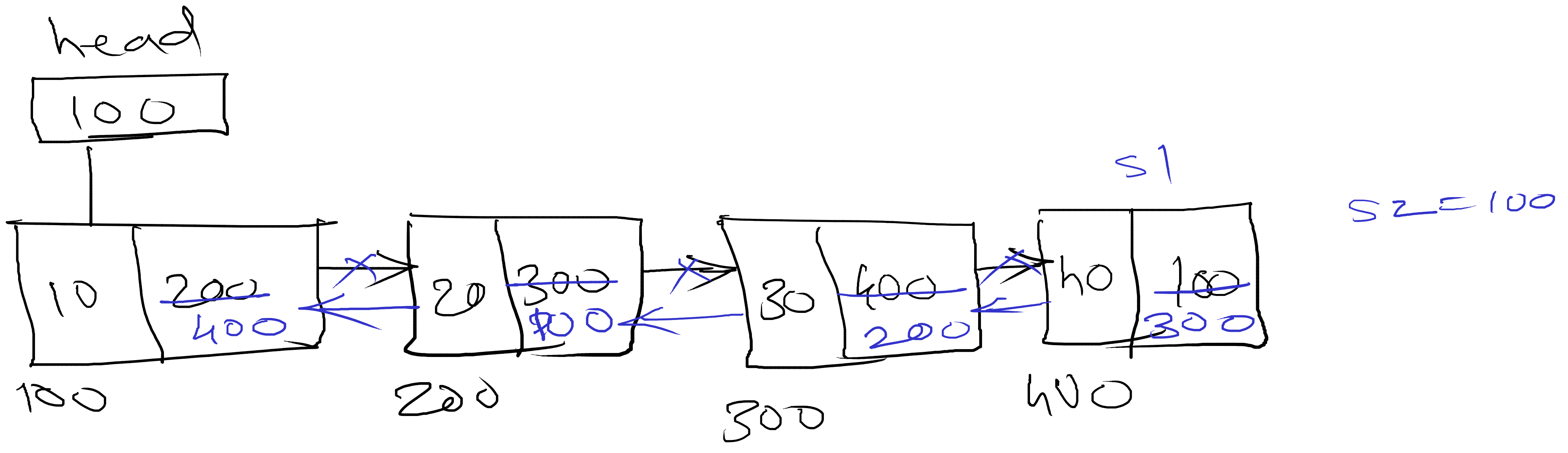
Linked List - Reverse



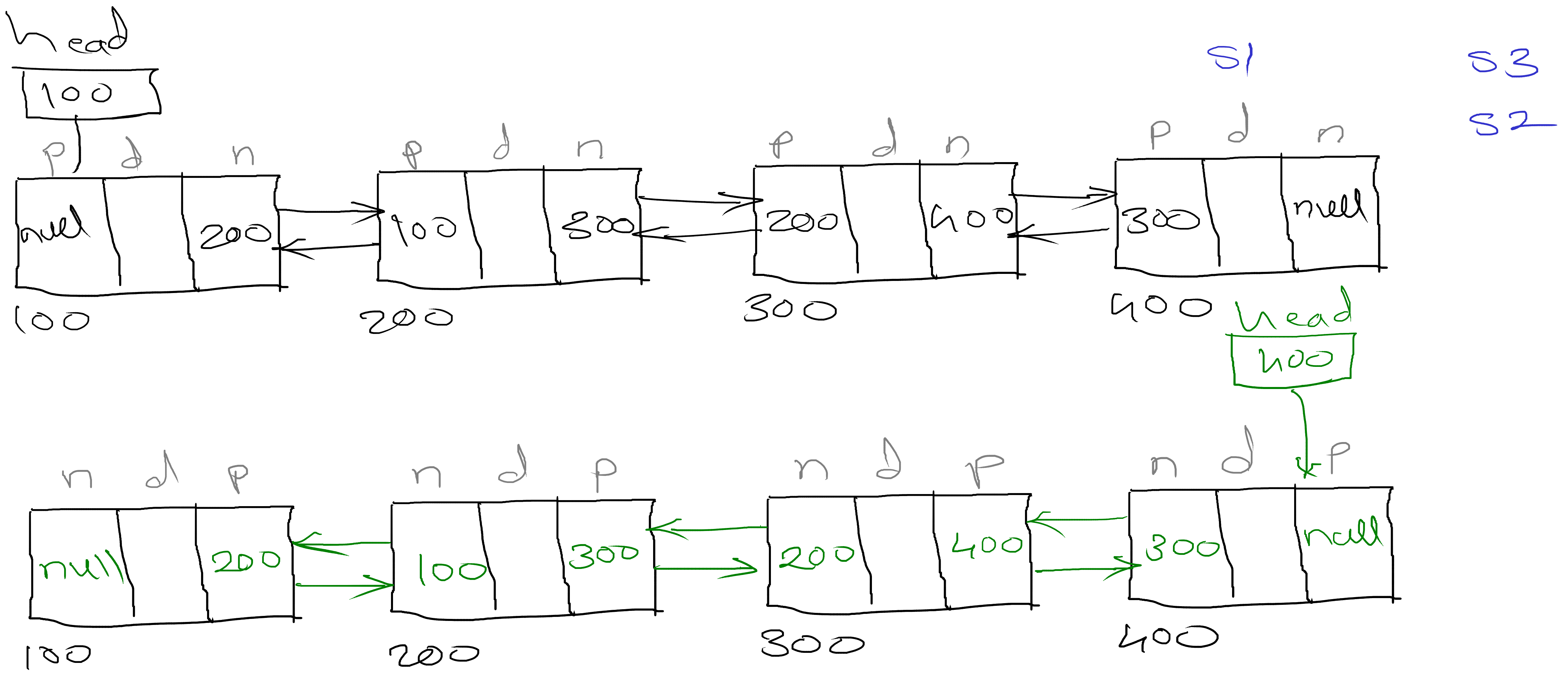


```
void reverseList( ){
        s1 = head; s2 = head.next
        while(s2 != null){
                s3 = s2.next;
                s2.next = s1;

                s1 = s2;
                s2 = s3;
        }
          head.next = null;
          head = s1;
}
```

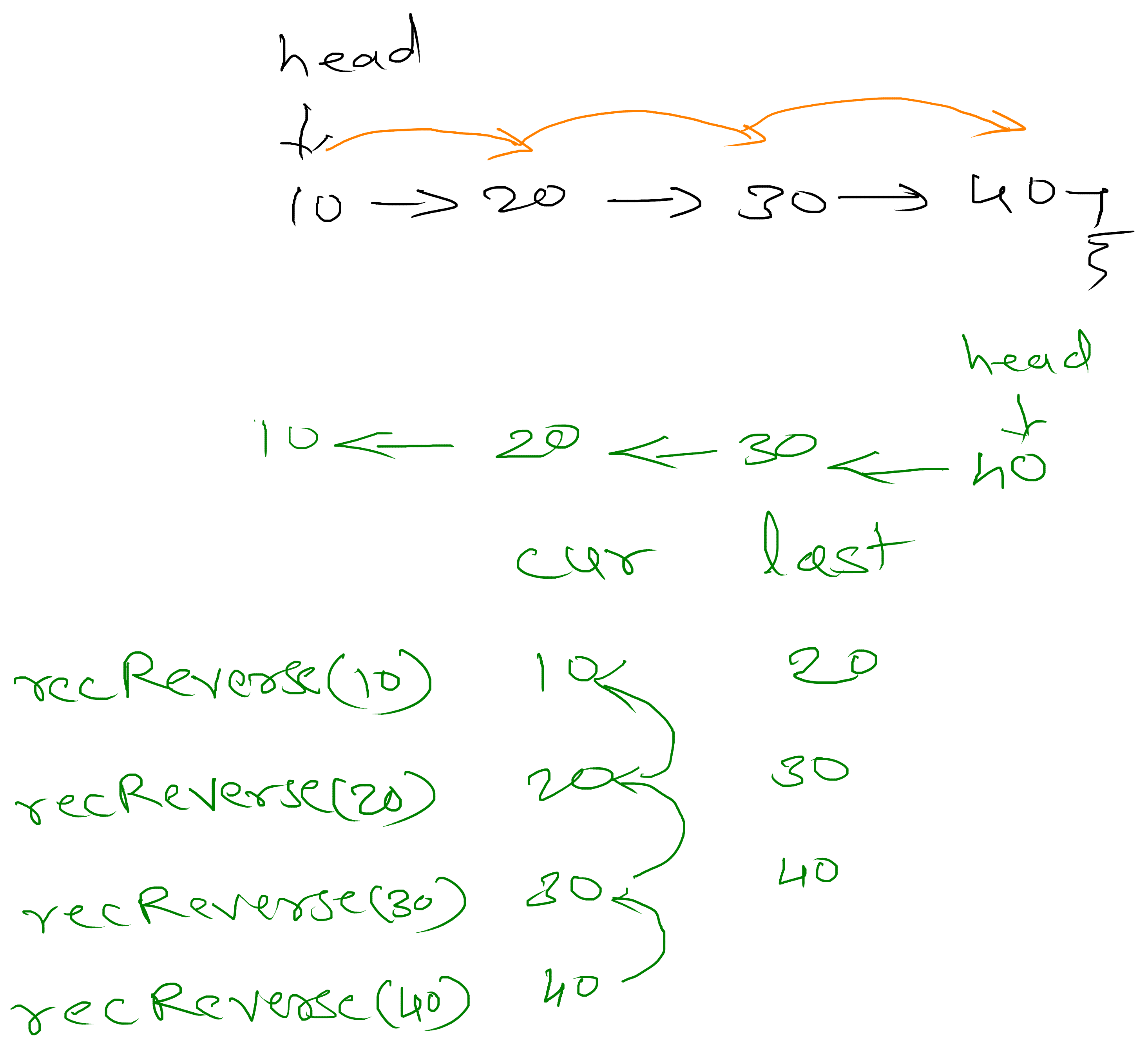


```
void reverseList( ) {
    s1=head , s2=s1.next;
    while(s2!=head) {
        s3=s2.next;
        s2.next= s1;
        s1=s2;
        s2=s3;
    }
    head.next=s1;
    head=s1;
}
```



```
void reverseList( ) {
    s1=head; s2 = s1.next;
    while(s2!=null){
        s3=s2.next;
        s2.next = s1;
        s1.prev = s2;
        s1=s2;
        s2=s3;
    }
    head.next=null;
    s1.prev=null;
    head=s1;
}
```

Reverse singly linked list using recursion.



```
Node recReverse(Node cur){
    if(cur.next==null){
        head=cur;
        cur.next=null;
        return cur;
    }
    last= recReverse(cur.next);
->  last.next = cur;
    cur.next = null;
    return cur;
}
```

Sort the singly linked list.

$$\begin{tikzcd}
\text{head} \arrow[d] & & & & \\
22 \arrow[r] & 55 \arrow[r] & 33 \arrow[r] & 44 \arrow[r] & 11
\end{tikzcd}$$

```
void selectionSort( ){
    Node i, j;
    for(i=head ; i != null ; i = i.next){
        for(j=i.next; j!=null; j=j.next){
            if(i.date > j.data)
                swap(i.data, j.date);

        }
}
```

Sort the singly linked list.

$$\begin{tikzcd}
\text{head} \arrow[d] & & & & \\
22 \arrow[r] & 55 \arrow[r] & 33 \arrow[r] & 44 \arrow[r] & 11
\end{tikzcd}$$

```
void  bubbleSort(   ){
    for(i=head; i.next !=null ; i=i.next){
        for(j=head; j.next.next !=null; j=j.next){
            if(j.data > j.next.data)
                swap(j.data, j.next.data)

        }

    }
```

Check if linked list is palindrome.

1. traverse list and push all elements on stack.
2. traverse list elements and pop element from stack one by one, compare them. if all elements are same then list is palindrome else not palindrome.

itrs = 2n

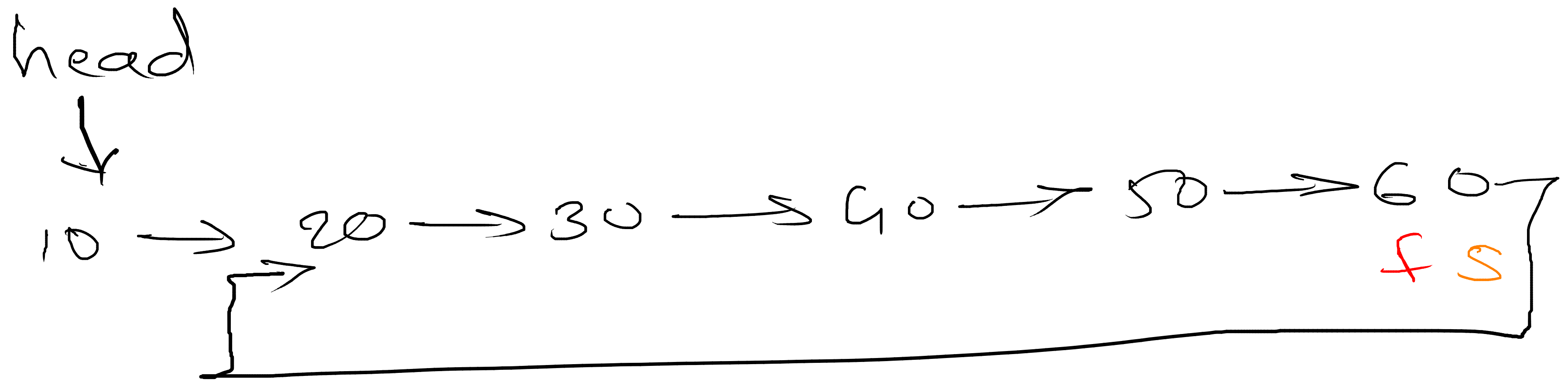
time = O(n)

aux space = O(n) stack.

```
booleam isPalindrome() {
    Stack<Integer> s = new Stack<>();
    for(trav=head; trav != null ; trav=trav.next)
            s.push(trav.data);
     for(trav=head; trav != null ; trav=trav.next){
          if(trav.data != s.pop())
               return false;   //not palindrome

     }
     return true;
}
```

Check if linked list contains a loop.



```
boolean hasloop() {
        s=f=head;
        while(f!=null && f.next != null) {
                s=s.next;
                f=f.next.next;
                if (f==s)
                    return true;

        }
        return false;
}
```

BST - Add Node

```
//1. Create node with given value
//2. if tree is empty
    //a. add newnode into root itself
//3. if tree is not empty
    //3.1 create trav referance and start at root
    //3.2 if value is less than current node data
        //3.2.1 if left of current node is empty
            //3.2.1.1 add node into left of current node
        //3.2.2 if left of current node is not empty
            //3.2.2.1 go on left side of current
    //3.3 if value is greater or equal than current node data
        //3.3.1 if right of current node is empty
            //3.3.1.1 add node into right of current node
        //3.3.2 if right of current node is not empty
            //3.3.2.1 go on right side of current
    //3.4 repeat step 3.2 and 3.3 till node is not added into tree
```