Remove duplicates from sorted LL.

① $2 \to 2 \to 4 \to 5 \Rightarrow 2 \to 4 \to 5$

t =

```
while (t.next != null)
{   if (t.data == t.next.data)
        { t.next = t.next.next }
    else
        t = t.next;
}
```

---

② Insert in sorted

5

1 14 20 21 25 ⇒ 1 7 14 7 29

f(key)

t

① (head == null)
nn = head;

② head.data > nn.data
{ Insert at beg.
}

③ while (t.next != null
&& t.next.data < nn.data)
t = t.next

else
nn.next = t.next
t.next = nn

③ Remove duplicates from Unsorted LL.

1st → $O(n^2)$ → two loops ✗

2nd → $O(n\log n)$ → sort & do as 2

(But order will be change for output)

Eg. 6 1 2 6 3 4 → 6 1 2 3 4 ✓

But it will come 1 2 3 4 6

3rd: hashing.

prev = null   6 1 2 6 3 4
curr head

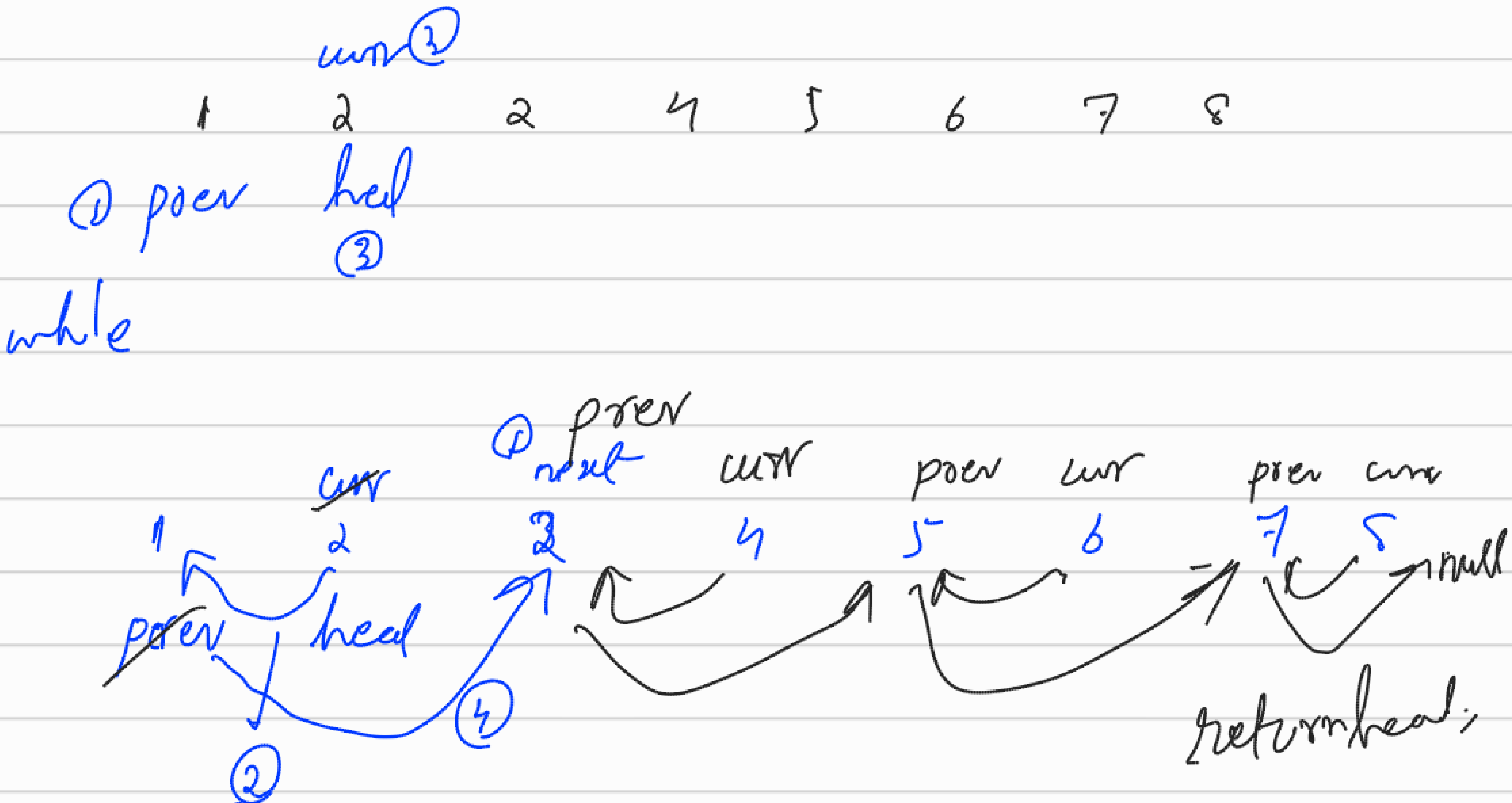
| 6 1 2 3 4 |

```
HashSet<Integer> hs = new HashSet<>();
Node curr = head;
Node prev = null;

while(curr != null)
{   int currval = curr.data;

    if (hs.contains(currval))
      { prev.next = curr.next }
    else
      { hs.add(currval);
        prev = curr;
}
```

```
        curr = curr.next;
    }
    return head;
```

④ Pairwise swap elements in LL.

head
1 2   2 4   5 6   7 8
2 1   4 2   6 5   8 7



```
③ if (next == null || next.next == null)
     { prev.next = next(null) ;
       break; }

④ else  prev.next = next.next
```

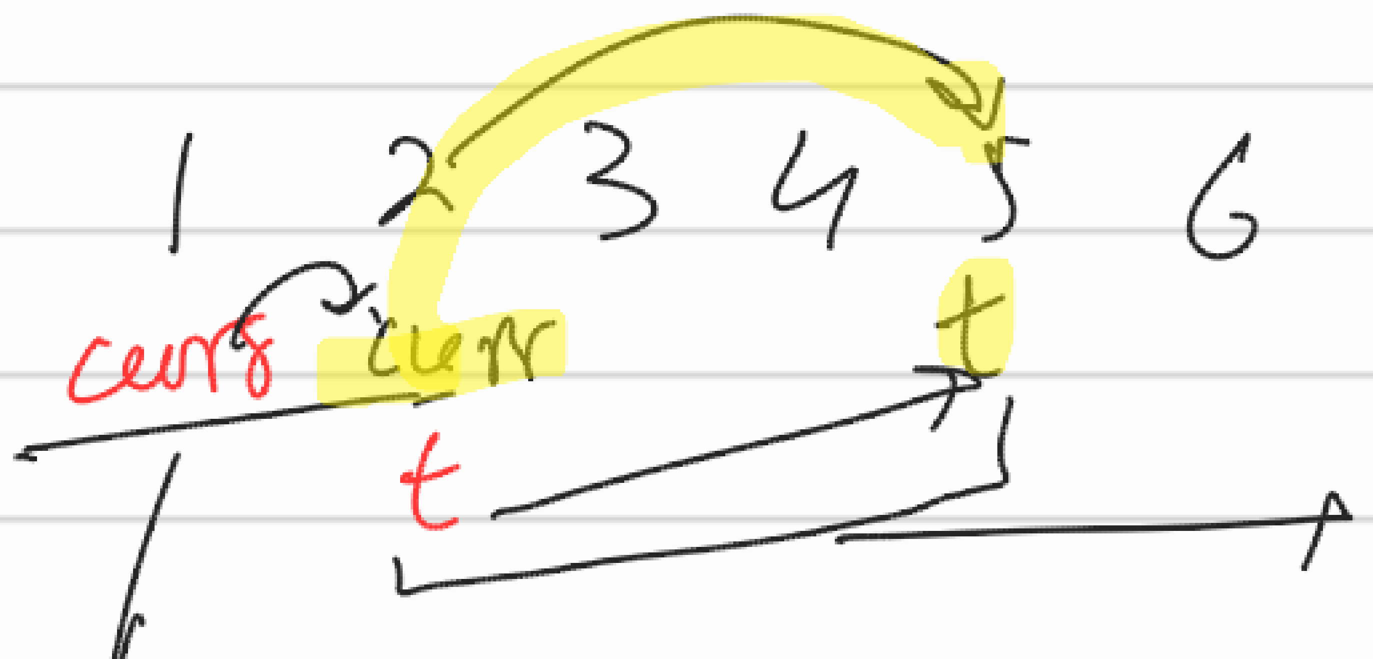
Full code:

```
if (head == null || head.next == null)
     { return head; }
```

```
Node prev= head;
Node curr= head.next;

head= curr                    (changing head to 2nd elem)

while (true)
{ ① Node next = curr.next;
  ②    curr.next = prev;
  ③  if ( next==null || next.next== null)
         {   prev.next= next (≤ is null)
               break;
         }
  ④ else{  prev.next= next.next; }

Iterate   prev= next;
           curr= prev.next; or next.next
}
   return head;
```

⑤. Delet N Node after M nodes.

8
2 2 → After 2 Node, delete 2.
1 2 ~~3 4~~ 5 6 ~~7 8~~ => 1 2 5 6



for (i=1; i<=N; i++)

→ for loop (i=1 ; i< m ; i++)

Code:

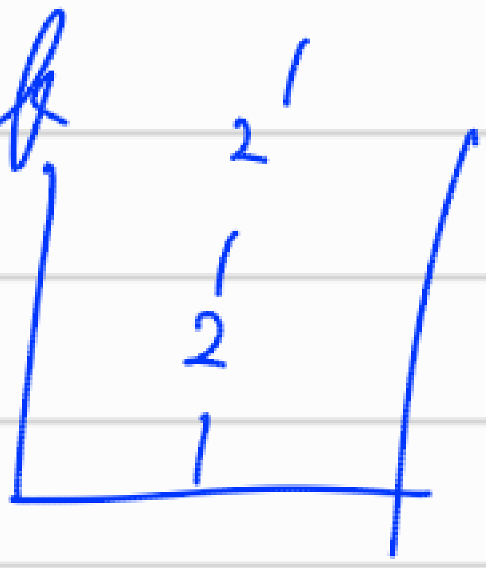
```
Node curr= head;
Node t= head;
  while (curr != null)
  {
    for (int i=1; i< m; i++)
      { curr= curr.next;
      }

      if curr == null
        { return
        }

    t= curr.next
    for( int i=1 ; i<= n ; i++)
      { t= t.next; }

      curr.next = t
        curr = t
  }
}
```

[traversed till curr
Eg:
6 1
1 2 3 4 5 6

---

⑥ Palindrome:

1→2→1→2→1 ⟹ Stack
t_1



traversing stack till empty
if (pop () == t_1.data)
  p = true

```
        else
            break;
```

```
      boolean is Palindrome (Node hea)
   {  boolean p= true;

      Stack<Integer> sc = new Stack();
          Node temp= head;
          while ( temp != null)
            {
               sc.push(temp.data);
                temp = temp.data;
             }
          Node t1= head
           while( t1 != null)
         {  if(sc.pop()== t1.data)
               {   p= true;
                     t1= t1.next;
                }
               else
                {   p= false
                        break;
                 }
            }
               return p;
   }
```