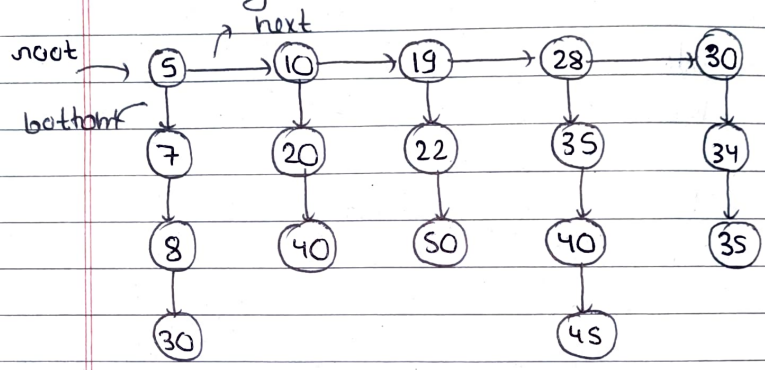
Date _____ Page _____

Day - 126

Linked List - 10

* **Flattening a LL:**



Sorted

⇒ We have to make singly Linked List

⇒ So, first, we merge the first two linked lists then remaining one by one.

Code for merging two LL

```
Node * merge( Node * head1 , Node *head2){
    Node * head = new Node(0);
    Node * tail = head;
    while( head1 && head2){
        if( head1 → data < head2 → data){
            tail → bottom = head1;
            head1 = head1 →
                bottom;
```

Date _____ Page _____

```
            Tail -> = Tail -> bottom;
            Tail -> bottom = NULL;
        }else{
            Tail -> bottom = head2;
            head2 = head2 -> bottom;
            tail = tail -> bottom;
            tail -> bottom = NULL;
        }
    }
    if(head1)
        tail -> bottom = head1;
    else
        tail -> bottom = head2;
    tail = head;   delete tail;
    return head;
}
```

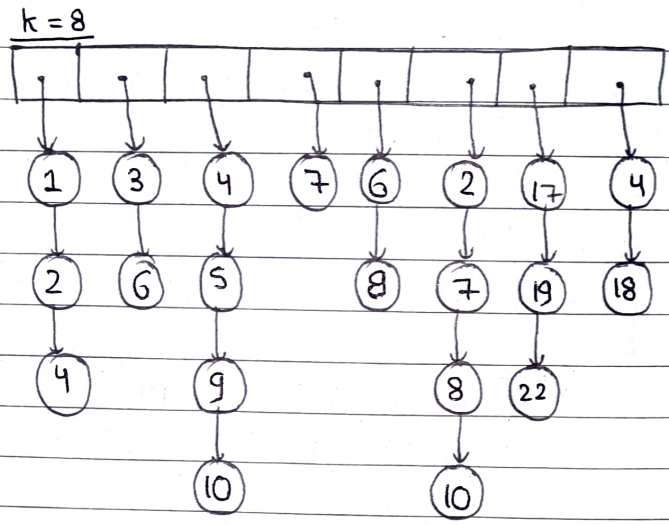
Code

```
Node * head1, *head2, *head3;
while(root -> next){
        head1 = root;
        head2 = root -> next;
        head3 = root -> next -> next;
        head1 -> next = NULL;
        head2 -> next = NULL;
        root = merge(head1, head2);
        root -> next = head3;
}
return root;
```

Date _____ Page _____

T.C. → $O(n^2m)$
S.C. → $O(1)$

* Merge k Sorted Linked List:



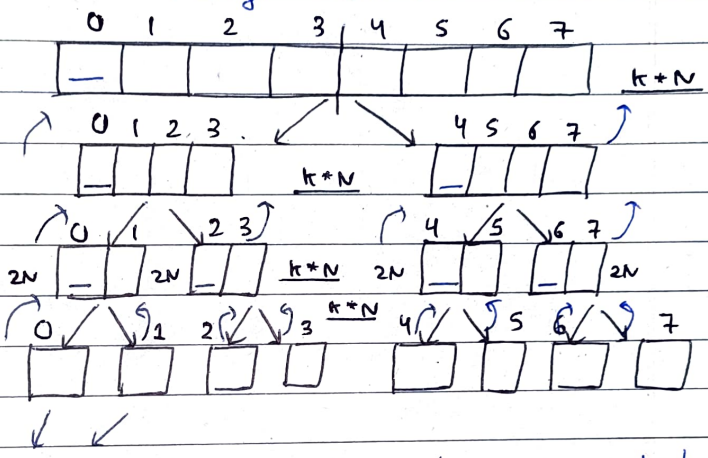
⇒ We have to generate a sorted LL.

⇒
```
Node *head = arr[0];
for (int i=1; i<k; i++){
    head = merge(head, arr[i]);
}
return head;
```

T.C. → $O(k^2N)$

Date _____ Page _____

⇉ To get more optimized way — we use merge sort.



Now we will sort these two LL and store in 0 index.

Code

```
mergeSort ( Node * arr[], int start,
                int end){
    if(start >= end)
        return;
    int mid = start + (end - start)/2;
    mergesort (arr, start, mid);
    mergesort (arr, mid+1, end);
arr[start]= merge (arr[start], arr[mid+1]);
    }
```

T.C. → O(nk log k)

S.C. → O(k)