MERGE K SORTED LINKED LISTS

☆ In this problem, we are given K sorted linked lists and our job is to return a new sorted linked list which merged existing linked lists.

Bruteforce solution is to iterate thro-ugh each list, add its elements to an array, sort the array and then convert it back to a linked list.

A better Solution is to group these K lists into pairs and use the merge two sorted linked lists repeated-ly.

```
Node * mergeKSortedLL(vector<Node*> &l){
    int K = l.size();
    Node * head = l[0];
    for(int i=1; i<K; i++) {
        head = merge2Lists(head, l[i]);
    }
    return head;
}
```

Time Complexity here is

$$N + 2N + 3N \cdots KN$$

$$N(1 + 2 + 3 \cdots)$$

$$\frac{NK(K+1)}{2} \approx O(N^3)$$

Optimal Solution involves use of priority queue (which implements min-heap, i.e. returning minimum value). We store the heads of each linked list and its index in a priority queue. We redeem the minimum, attach it to dummy node and repeat.

```
Node * mergeKSortedLL(vector<Node*> l){
    int K = l.size();
    Node* dN = new Node(-1);
    priority_queue<pair<int, Node*>> pq;
    for(int i=0; i<K; i++) {
        pq.push({l[i]->val, l[i]});
    }
    Node * temp = dN;
    while(!pq.empty()) {
        auto it = pq.top();
        pq.pop();
        if(it.second->next != nullptr) {
            pq.push({it.second->next->val,
                     it->second->next});
        }
        temp->next = it.second;
        temp = temp->next;
    }
    return dN->next;
}
```