MCS 253P Quiz 4

Name: S. ASWIN **UciNetID:** saswin

1) (80 pts) Given the **head** of a singly linked list and an integer **k**, split the linked list into **k** consecutive linked list parts. The length of each part should be as equal as possible: no two parts should have a size differing by more than one. This may lead to some parts being null. The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.
Return an array of the **k** parts.

(20 pts) Please include **2 testcases.**

Example:
Input: head = [1,2,3], k = 5
Output: [[1],[2],[3],[],[]]

Logic: T1: head = [1,2,3,4,5,6], K = 4
ans = (1,2), (3,4), (5), (6) → [1, 3, 5, 6]

T2; head = [1,2,3,4] K = 5
ans = (1,) (2) (3) (4) (NULL)

1. Find the length of linked list
2. If K > n, add NULL values at the end of answer
3. Compute r = n % k. If r != 0 the first r linked lists will have size equal and '1' more than K − r lists
4. Generate each linked lists & add their front node to answer.
5. Time complexity : O(n) { Referencing each node only once }
   Space complexity : O(n)

```
int findLength(Node *root){ Node *current = root;
    int count = 0;
    while(root != NULL){ count++;
        root = root->next;
    }
    return count;
}

Node *generateLinkedList(Node *&head, int size){ // pass by reference
    Node *newLLhead = NULL, *newLLtail = NULL;
    while(size--){ Node *newNode = new Node(head->data);
        if(newLLhead == NULL){
            newLLhead = newNode; newLLtail = newNode;
        }
        else{ newLLtail->next = newNode;
            newLLtail = newNode;
        }
        head = head->next;
    }
}

vector<Node*> splitK(Node *head){
    int n = findLength(head);
    int r = n % k;
    vector<Node*> ans;
    while(head != NULL){
        if(r) size = k+1; else size = k;
        if(k > n) size = 1;
        Node *newListHead = generateLinkedList(
                head, size);
        ans.append(newListHead);
        r--;
    }
    int rem = k - n;
    while(rem--)
        ans.append(NULL);
    return ans;
}
```