# Split Linked List Alternatingly

| | |
|---|---|
| solved by | Senan |
| Platform | GeeksForGeeks |
| difficulty | Easy |
| tags | Linked List |
| language | C++ |
| solved on | @17/10/2024 |
| link | https://www.geeksforgeeks.org/problems/split-singly-linked-list-alternatingly/1 |
| Completion | ✓ |

## Intuition

To split a linked list alternately into two separate lists, we can iterate over the original linked list and, on each iteration, alternate the nodes between the two resulting lists. One list will take the nodes at odd positions, while the other will take those at even positions.

## Approach

1. We initialize two dummy nodes, `first` and `second`, to represent the starting points of our two resultant lists.
2. We traverse the original linked list using a pointer `temp`, alternating between appending nodes to the two lists.
3. To keep track of which list should receive the next node, we use a boolean flag `change`, which toggles after each node is processed.
4. Once the traversal is done, we need to set the `next` pointer of the last node in both lists to `NULL` to signify the end of the lists.
5. Finally, we return the two lists, excluding the dummy nodes.

## Complexity

### Time Complexity:

- **O(n)**, where `n` is the number of nodes in the linked list. We only traverse the list once.

### Space Complexity:

- **O(1)** for extra space (excluding the space for the output lists). We are not using any additional space that grows with input size except the output lists.

## Code

```
class Solution {
  public:
    // Function to split a linked list into two lists alternately
    vector<Node*> alternatingSplitList(struct Node* head) {
        Node* first = new Node(-1);
        Node* second = new Node(-1);

        Node* marker1 = first;
        Node* marker2 = second;

        Node* temp = head;
        bool change = true;

        while(temp) {
            if(change) {
                first->next = temp;
                first = temp;
            } else {
                second->next = temp;
                second = temp;
            }
            change = !change;
            temp = temp->next;
        }

        first->next = NULL;
        second->next = NULL;

        first = marker1->next;
        second = marker2->next;

        delete marker1;
        delete marker2;

        return {first, second};
    }
};
```