Find anagrams in linked list

Accuracy: 52.94% Submissions: 34 Points: 50

Given a **linked list of characters** and a **string S**. Return all the anagrams of the string present in the given linked list. In case of overlapping anagrams choose the first anagram from left (for more clear explanation, look at the first example).

Example 1:

```
Input: a -> b -> c -> a -> d -> b -> c -> a
S = bac
Output: [a -> b -> c, b -> c -> a]
Explanation: In the given linked list,
there are three anagrams:
1. a -> b -> c -> a -> d -> b -> c -> a
2. a -> b -> c -> a -> d -> b -> c -> a
3. a -> b -> c -> a -> d -> b -> c -> a
But in 1 and 2, a -> b -> c and b -> c-> a
are ovelapping. So we take a -> b -> c as it
comes first from left. So the output is:
[a->b->c,b->c->a]
```

Example 2:

```
Input: a -> b -> d -> c -> a
S = bac
Output: []
```

Explanation: Since there are no anagrams, we return an empty array.

Your Task:
You don't need to read input or print anything. Your task is to complete the function **findAnagrams()** which takes head node of the linked list and a string S as input parameters and returns an array of linked list.

Expected Time Complexity: O(N)
Expected Auxiliary Space: O(1)

C++ (g++ 5.4) Test against custom input

```
1  // } Driver Code Ends
41 /*
42
43 Definition for singly Link List Node
44 struct Node
45 {
46     char data;
47     Node* next;
48     Node(char x) {  data = x;  next = NULL; }
49 };
50
51 You can also use the following for printing the link list.
52 printList(Node* node);
53 */
54
55 class Solution {
56   public:
57     vector<Node*> findAnagrams(struct Node* head, string s) {
58         // code here
59     }
60 };
61
62
63  // } Driver Code Ends
```

Compile & Run Submit