Identical Linked Lists

Two Linked Lists are identical when they have same data and arrangement of data is also same. For example Linked lists a (1->2->3) and b(1->2->3) are identical. . Write a function to check if the given two linked lists are identical.

Recommended: Please solve it on "*PRACTICE*" first, before moving on to the solution.

Method 1 (Iterative)

To identify if two lists are identical, we need to traverse both lists simultaneously, and while traversing we need to compare data.

C++ C Java Python3 C#

```
// An iterative C++ program to check if
// two linked lists are identical or not
#include<bits/stdc++.h>
using namespace std;

/* Structure for a linked list node */
struct Node
{
    int data;
    struct Node *next;
};

/* Returns true if linked lists a and b
are identical, otherwise false */
bool areIdentical(struct Node *a,
                  struct Node *b)
{
    while (a != NULL && b != NULL)
    {
        if (a->data != b->data)
            return false;

        /* If we reach here, then a and b are
        not NULL and their data is same, so
        move to next nodes in both lists */
        a = a->next;
        b = b->next;
    }

    // If linked lists are identical, then
    // 'a' and 'b' must be NULL at this point.
    return (a == NULL && b == NULL);
}

/* UTILITY FUNCTIONS TO TEST fun1() and fun2() */
/* Given a reference (pointer to pointer) to the
head of a list and an int, push a new node on the
front of the list. */
```

```
void push(struct Node** head_ref, int new_data)
{
    /* allocate node */
    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));

    /* put in the data */
    new_node->data = new_data;

    /* link the old list off the new node */
    new_node->next = (*head_ref);

    /* move the head to point to the new node */
    (*head_ref) = new_node;
}

// Driver Code
int main()
{
    /* The constructed linked lists are :
    a: 3->2->1
    b: 3->2->1 */
    struct Node *a = NULL;
    struct Node *b = NULL;
    push(&a, 1);
    push(&a, 2);
    push(&a, 3);
    push(&b, 1);
    push(&b, 2);
    push(&b, 3);

    if(areIdentical(a, b))
        cout << "Identical";
    else
        cout << "Not identical";

    return 0;
}

// This code is contributed
// by Akanksha Rai
```

Output:

```
Identical
```

Method 2 (Recursive)

Recursive solution code is much cleaner than the iterative code. You probably wouldn't want to use the recursive version for production code however, because it will use stack space which is proportional to the length of the lists

C++ C Java Python3 C#

```cpp
// A recursive C++ function to check if two linked
// lists are identical or not
bool areIdentical(Node *a, Node *b)
{
    // If both lists are empty
    if (a == NULL && b == NULL)
    return true;

    // If both lists are not empty, then data of
    // current nodes must match, and same should
    // be recursively true for rest of the nodes.
    if (a != NULL && b != NULL)
    return (a->data == b->data) &&
            areIdentical(a->next, b->next);

    // If we reach here, then one of the lists
    // is empty and other is not
    return false;
}

//This is code is contributed by rathbhupendra
```

Time Complexity: O(n) for both iterative and recursive versions. n is the length of the smaller list among a and b.

Please write comments if you find the above codes/algorithms incorrect, or find better ways to solve the same problem.