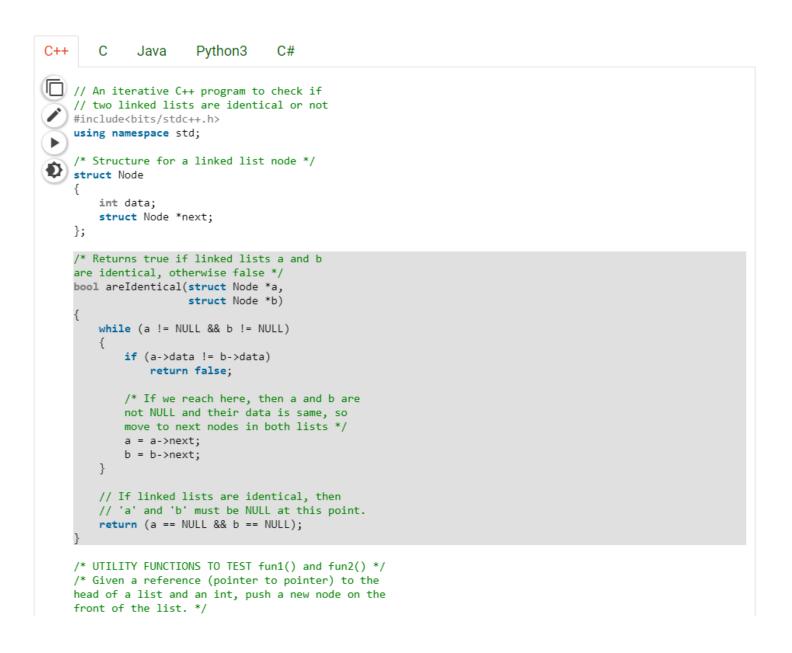
## Identical Linked Lists

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.



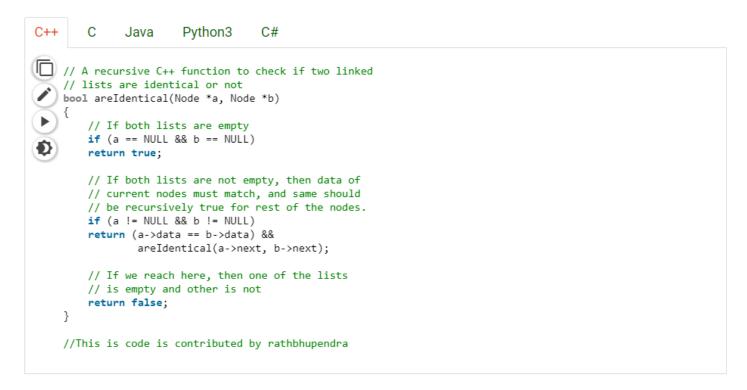
```
void push(struct Node** head_ref, int new_data)
    /* allocate node */
    struct Node* new_node =
        (struct Node*) malloc(sizeof(struct Node));
    /* put in the data */
    new_node->data = new_data;
    /* link the old list off the new node */
    new_node->next = (*head_ref);
    /* move 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";</pre>
    else
        cout << "Not identical";</pre>
    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



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.