Check if a linked list of strings forms a palindrome

Given a linked list handling string data, check to see whether data is palindrome or not?

For example,

```
Input  : a -> bc -> d -> dcb -> a -> NULL
Output : True
String "abcddcba" is palindrome.

Output : a -> bc -> d -> ba -> NULL
Output : False
String "abcdba" is not palindrome.
```

We strongly recommend you to minimize your browser and try this yourself first.

The idea is very simple. Construct a string out of given linked list and check if the constructed string is palindrome or not.

C/C++

```
// Program to check if a given linked list of strings
// form a palindrome
#include <bits/stdc++.h>
using namespace std;

/* Link list node */
struct Node
{
    string data;
    Node* next;
};

// A utility function to check if str is palindrome
// or not
bool isPalindromeUtil(string str)
{
    int length = str.length();

    // Match characters from beginning and
    // end.
    for (int i=0; i<length; i++)
        if (str[i] != str[length-i-1])
            return false;

    return true;
}

// Returns true if string formed by linked
// list is palindrome
bool isPalindrome(Node *node)
{
    // Append all nodes to form a string
    string str = "";
    while (node != NULL)
    {
        str.append(node->data);
        node = node->next;
    }

    // Check if the formed string is palindrome
    return isPalindromeUtil(str);
}

// A utility function to print a given linked list
// A utility function to print a given linked list
void printList(Node *node)
{
    while (node != NULL)
    {
        cout << node->data << " -> ";
        node = node->next;
    }
    printf("NULL\n");
}

/* Function to create a new node with given data */
Node *newNode(const char *str)
{
    Node *new_node = new Node;
    new_node->data = str;
    new_node->next = NULL;
    return new_node;
}

/* Driver program to test above function*/
int main()
{
    Node *head = newNode("a");
    head->next = newNode("bc");
    head->next->next = newNode("d");
    head->next->next->next = newNode("dcb");
    head->next->next->next->next = newNode("a");

    isPalindrome(head)? printf("true\n"):
                        printf("false\n");

    return 0;
}
```

Java

```java
// Java Program to check if a given linked list of strings
// form a palindrome

import java.util.Scanner;

// Linked List node
class Node
{
    String data;
    Node next;

    Node(String d)
    {
        data = d;
        next = null;
    }
}

class LinkedList_Palindrome
{
    Node head;

    // A utility function to check if str is palindrome
    // or not
    boolean isPalidromeUtil(String str)
    {
        int length = str.length();

        // Match characters from beginning and
        // end.
        for (int i=0; i<length; i++)
            if (str.charAt(i) != str.charAt(length-i-1))
                return false;

        return true;
    }

    // Returns true if string formed by linked
    // list is palindrome
    boolean isPalindrome()
    {
        Node node = head;

        // Append all nodes to form a string
        String str = "";
        while (node != null)
        {
            str = str.concat(node.data);
            node = node.next;
        }

        // Check if the formed string is palindrome
        return isPalidromeUtil(str);
    }

    /* Driver program to test above function*/
    public static void main(String[] args)
    {
        LinkedList_Palindrome list = new LinkedList_Palindrome();
        list.head = new Node("a");
        list.head.next = new Node("bc");
        list.head.next.next = new Node("d");
        list.head.next.next.next = new Node("dcb");
        list.head.next.next.next.next = new Node("a");

        System.out.println(list.isPalindrome());

    }
}
// This code has been contributed by Amit Khandelwal
```

Python

```python
# Python program to check if given linked list of strings
# form a palindrome

# Node class
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # A utility function to check if str is palindrome
    # or not
    def isPalindromeUtil(self, string):
        return (string == string[::-1])

    # Returns true if string formed by linked list is
    # palindrome
    def isPalindrome(self):
        node = self.head
        # Append all nodes to form a string
        temp = []
        while(node is not None):
            temp.append(node.data)
            node = node.next
        string = "".join(temp)
        return self.isPalindromeUtil(string)

    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next


# Driver program to test above function
llist = LinkedList()
llist.head = Node('a')
llist.head.next = Node('bc')
llist.head.next.next = Node("d")
llist.head.next.next.next = Node("dcb")
llist.head.next.next.next.next = Node("a")
print "true" if llist.isPalindrome() else "false"

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
true
```