Delete duplicate-value nodes from a sorted linked list

This challenge is part of a tutorial track by MyCodeSchool

You are given the pointer to the head node of a sorted linked list, where the data in the nodes is in ascending order. Delete nodes and return a sorted list with each distinct value in the original list. The given head pointer may be null indicating that the list is empty.

Example

$head$ refers to the first node in the list $1 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow 3 \rightarrow NULL$.

Remove 1 of the 2 data values and return $head$ pointing to the revised list $1 \rightarrow 2 \rightarrow 3 \rightarrow NULL$.

Function Description

Complete the *removeDuplicates* function in the editor below.

removeDuplicates has the following parameter:

- *SinglyLinkedListNode pointer head:* a reference to the head of the list

Returns

- *SinglyLinkedListNode pointer:* a reference to the head of the revised list

Input Format

The first line contains an integer t, the number of test cases.

The format for each test case is as follows:

The first line contains an integer n, the number of elements in the linked list.
Each of the next n lines contains an integer, the $data$ value for each of the elements of the linked list.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq list[i] \leq 1000$

Sample Input

```
STDIN   Function
-----   --------
1       t = 1
5       n = 5
1       data values = 1, 2, 2, 3, 4
2
2
```

```
3
4
```

Sample Output

```
1 2 3 4
```

Explanation

The initial linked list is: $1 \rightarrow 2 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow NULL$.

The final linked list is: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow NULL$.

Contest ends in 3 days

Submissions: 45
Max Score: 10
Difficulty: Easy

Rate This Challenge:
☆☆☆☆☆

Download problem statement

Download sample test cases

Suggest Edits

Collapse

Java 15

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class SinglyLinkedListNode {
    public int data;
    public SinglyLinkedListNode next;

    public SinglyLinkedListNode(int nodeData) {
        this.data = nodeData;
        this.next = null;
    }
}

class SinglyLinkedList {
    public SinglyLinkedListNode head;
    public SinglyLinkedListNode tail;

    public SinglyLinkedList() {
        this.head = null;
        this.tail = null;
    }

    public void insertNode(int nodeData) {
        SinglyLinkedListNode node = new SinglyLinkedListNode(nodeData);
```

```java
        if (this.head == null) {
            this.head = node;
        } else {
            this.tail.next = node;
        }

        this.tail = node;
    }
}

class SinglyLinkedListPrintHelper {
    public static void printList(SinglyLinkedListNode node, String sep, BufferedWriter
bufferedWriter) throws IOException {
        while (node != null) {
            bufferedWriter.write(String.valueOf(node.data));

            node = node.next;

            if (node != null) {
                bufferedWriter.write(sep);
            }
        }
    }
}

class Result {

    /*
     * Complete the 'removeDuplicates' function below.
     *
     * The function is expected to return an INTEGER_SINGLY_LINKED_LIST.
     * The function accepts INTEGER_SINGLY_LINKED_LIST llist as parameter.
     */

    /*
     * For your reference:
     *
     * SinglyLinkedListNode {
     *     int data;
     *     SinglyLinkedListNode next;
     * }
     *
     */


public static SinglyLinkedListNode  removeDuplicates(SinglyLinkedListNode  head) {
  // This is a "method-only" submission.
  // You only need to complete this method.

    if(head==null)
        return null;

   SinglyLinkedListNode  temp=head.next;
    SinglyLinkedListNode  prev=head;
    while(temp!=null)
        {

         if(prev.data==temp.data)
             {
             prev.next=temp.next;
             temp.next=null;
             temp=prev.next;
         }
        else
            {
              prev=temp;
              temp=temp.next;

        }
    }
```

```java
        return head;
    }

}

public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter(System.getenv("OUTPUT_PATH")));

        int t = Integer.parseInt(bufferedReader.readLine().trim());

        IntStream.range(0, t).forEach(tItr -> {
            try {
                SinglyLinkedList llist = new SinglyLinkedList();

                int llistCount = Integer.parseInt(bufferedReader.readLine().trim());

                IntStream.range(0, llistCount).forEach(i -> {
                    try {
                        int llistItem = Integer.parseInt(bufferedReader.readLine().trim());

                        llist.insertNode(llistItem);
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }
                });

                SinglyLinkedListNode llist1 = Result.removeDuplicates(llist.head);

                SinglyLinkedListPrintHelper.printList(llist1, " ", bufferedWriter);
                bufferedWriter.newLine();
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        });

        bufferedReader.close();
        bufferedWriter.close();
    }
}
```

Line: 79 Col: 15

Upload Code as File ☐ Test against custom input Run Code Submit Code

Testcase 0 ✓ Testcase 1 ✓

Congratulations, you passed the sample test case.

Click the **Submit Code** button to run your code against all the test cases.

Input (stdin)

```
1
5
1
2
2
3
4
```

Your Output (stdout)

```
1 2 3 4
```

Expected Output

```
1 2 3 4
```