Hello Data Structures! If you made it till this stage, give yourself a pat on the back, I am proud of you! This lab is optional and will be used to replace the grade of your poorest lab performance during the semester ;). If you think you did good in all the labs, Salute...feel free to leave the session.

Choose only ONE of the problems to solve from the 3 problems below.

Problem 1

Given a linked list of integers, write a function getUnique that removes all duplicates elements in the linked list and returns the new linked list of unique elements (The order does not matter).

Example:

Input: 1->2->3->1->2
Output: 1->2->3

Problem 2

Given a string input str and a list of words to be ignored toIgnore, write a function getMostCommonWord that takes str and toIgnore as arguments and returns the most frequent word that is not in the toIgnore list. (Hint: Use a hashmap<String, Integer> for recording the frequency of all the words.)

Example:

Input:
str = "I like queues but queues do not like me"
toIgnore = ["queues"]
Output: "like"

Problem 3

Write a function called findUniqueIndex that takes a string str as input, finds the first non-repeating character in str, and return the index of the character in str. If all characters are non-unique, the function should return -1.

Example:

Input:
str = "CSC2720"
Output:
1
Explanations:
S is the first unique character in str and it is located at index 1.

Input:
str = "toto"
Output:
-1
Explanations:
All the characters appear at least twice in str, so no character in unique.
public class Node {
    int item;
    Node next;

    Node(int d) {
        item = d;
        next = null;
    }
}

class ExtraLab {
    public static void main(String[] args) {
        // PROBLEM 1
        System.out.println("PROBLEM 1");
        Node head = new Node(1);
        head.next = new Node(2);
        head.next.next = new Node(3);
        head.next.next.next = new Node(1);
        head.next.next.next.next = new Node(2);

        head = getUnique(head);
        printLinkedList(head);

        // PROBLEM 2
        System.out.println("PROBLEM 2");
        String str = "I like queues but queues do not like me";
        ArrayList<String> toIgnore = new ArrayList<String>();
        toIgnore.add("stacks");
        System.out.println(getMostCommonWord(str, toIgnore)); // should return "like"

        // PROBLEM 3
        System.out.println("PROBLEM 3");
        str = "CSC2720";
        System.out.println(findUniqueId(str)); // Should return 1
        str = "toto";
        System.out.println(findUniqueId(str)); // Should return -1
    }

    public static Node getUnique(Node head) {
        // PROBLEM 1: INSERT CODE HERE
        return null;
    }

    public static int findUniqueId(String str) {
        // PROBLEM 2: INSERT CODE HERE
        return 0;
    }

    public static String getMostCommonWord(String str, ArrayList<String> list) {
        // PROBLEM 3: INSERT CODE HERE
        return null;
    }

    public static void printLinkedList(Node head) {
        for (Node cur = head; cur != null; cur = cur.next) {
            System.out.print(cur.item + " ");
        }
        System.out.println();
    }
}