

CMSC132, Fall 2021, QUIZ #3 (DURATION: 30 MINUTES) – 30 pts

FIRSTNAME, LASTNAME (PRINT IN UPPERCASE):

STUDENT ID (e.g. 123456789):

INSTRUCTIONS:

Assume the code below with all necessary import statements.

```
public class StringList {

    private class Node {
        private String data;
        private Node next;

        private Node(String data) {
            this.data = data;
            next = null;
        }
    }

    private Node head;
    private Node tail;

    public StringList() {
        head = tail= null;
    }

    public StringList add(String data) {
        Node newNode = new Node(data);
        if(head==null)
        {
            newNode.next = head;
            head = newNode;
            tail = head;
        }
        else
        {
            tail.next = newNode ;
            tail =  tail.next;
        }

        return this;
    }

    public String toString() {
        String result = "\\\" ";
        Node curr = head;

        while (curr != null) {
            result += curr.data + " ";
            curr = curr.next;
        }

        return result + "\\\"";
    }

    public ArrayList<StringList> makeList(String headValue) {
        //Code here needs to make the first call to private void makeListAux method
    }

    //code for the private void makeListAux method - MUST BE RECURSIVE, NO LOOPS!!!
    //Can have up to 5 parameters
}
```

```

public class SampleDriver {

    public static void main(String[] args) {

        StringList myList = new StringList();

        myList.add("if").add("else").add("for").add("int").add("for");
        myList.add("for").add("cmsc132");
        myList.add("if").add("for").add("java");
        System.out.println(myList);

        System.out.println("ArrayList for first call");
        for(StringList s: myList.makeList("for"))
            System.out.println(s);

        System.out.println("ArrayList for second call");
        for(StringList s: myList.makeList("map")) //empty arrayList returned
            System.out.println(s);

    }

}

```

---

#### Driver Output

```

" if else for int for for cmsc132 if for java "
ArrayList for first call
" for int "
" for "
" for cmsc132 if "
" for java "
ArrayList for second call

```

---

`makeList` will return an ArrayList where each element in the ArrayList will reference a `StringList` object with the head node containing the `headValue` (argument passed in to `makeList`). `makeList` will make a call to `makeListAux` which will recursively traverse the current `StringList` object (i.e. caller of `makeList`) ignoring all values until it finds `headValue`. Once `headValue` is found, a new `StringList` is created and assigned as an element of the ArrayList using the process described below.

1. A new `StringList` object should be created and assigned to an element of the ArrayList. The head of this new list should point to a new node that contains `headValue` as its data.
2. All other nodes encountered while traversing the current `StringList` should be added as new nodes of this new `StringList`.
3. The process of adding nodes to this new `StringList` only stops if you run out nodes in the current `StringList` object or if the `headValue` is encountered again.

If the `headValue` is encountered again, the 3 steps above should occur again to make yet a new `StringList` object that will be assigned to a subsequent element of the ArrayList.

In the sample driver, the `headValue` is `for`. `if` and `else` from the current `StringList` (i.e. `myList`) object are ignored. The first `for` that is encountered causes the creation of a new `StringList` object being assigned to the ArrayList with a node with the value of `for` as the head of this new `StringList`. `int` is added as the next node of this new `StringList`. When `for` is encountered again, this cause a new `StringList` object to be created and assigned to the second element of the ArrayList. When `for` is encountered yet a third time, this causes a third `StringList` to be created and the reference is assigned as the third element of the ArrayList. `cmsc132` and `if` are added as nodes to this third `StringList`. Finally, encountering

for again, cause a new 4<sup>th</sup> `StringList` to be created with `for` followed by `java`. Since you are working with Strings, deep copies of the data are not necessary, but make sure each `StringList` has its own nodes (no sharing).

//Solution by Matthew Simmons

```
public ArrayList<StringList> makeList(String headValue) {
    ArrayList<StringList> lst = new ArrayList<>();
    makeList(headValue, lst, head, -1);
    return lst;
}

private void makeList(String headValue, ArrayList<StringList> lst, Node curr, int count) {
    if(curr != null) {
        if(curr.data.equals(headValue)) {
            count++;
            lst.add(new StringList());
        }
        if(count >= 0) { // Count not negative means the above has run
            lst.get(count).add(curr.data);
        }
        makeList(headValue, lst, curr.next, count);
    }
}
```