Copy List with Random Pointer (LeetCode 138 - Hard)

Problem Description

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a deep copy of the list.

List Node Class

```
public class RandomListNode {
	int label;
	RandomListNode next, random;

	RandomListNode(int x) {
		this.label = x;
	}
}
```

Solution 1

```
/*
 * Use HashMap.
 * Time Complexity - O(n)
 * Space Complexity - O(n) for HashMap.
 */
public static RandomListNode copyRandomList1(RandomListNode head) {
	if (head == null) {
		return head;
	}
	RandomListNode temp = head;
	HashMap<RandomListNode, RandomListNode> map = new HashMap<>();
	while (temp != null) {
		map.put(temp, new RandomListNode(temp.label));
		temp = temp.next;
	}

	temp = head;
	while (temp != null) {
		RandomListNode cur = map.get(temp);
		cur.next = temp.next == null ? null : map.get(temp.next);
		cur.random = temp.random == null ? null : map.get(temp.random);
		temp = temp.next;
	}

	return map.get(head);
}
```

Solution 2

```
/*
 * Time Complexity - O(n)
 * Space Complexity - O(1)
 */
public RandomListNode copyRandomList2(RandomListNode head) {
  if (head == null) {
            return head;
        }

        // 1. copy each node and append the new node after its father
        RandomListNode temp = head;
        while (temp != null) {
            RandomListNode newNode = new RandomListNode(temp.label);
            newNode.next = temp.next;
            temp.next = newNode;
            temp = newNode.next;
        }

        // 2. assign random pointer
        temp = head;
        while (temp != null) {
            RandomListNode newNode = temp.next;
            newNode.random = temp.random == null ? null : temp.random.next;
            temp = temp.next.next;
        }

        // 3. reconstruct the original list
        temp = head;
        RandomListNode newNode = head.next;
        RandomListNode rv = newNode;
        while (temp != null) {
            temp.next = temp.next == null ? null : temp.next.next;
            newNode.next = temp.next == null ? null : temp.next.next;
            temp = temp.next;
            newNode = temp == null ? null : temp.next;
        }

        return rv;
    }
```