Quiz #8: Linked Lists INF-42, Winter 2011

Name _____

Lab _____

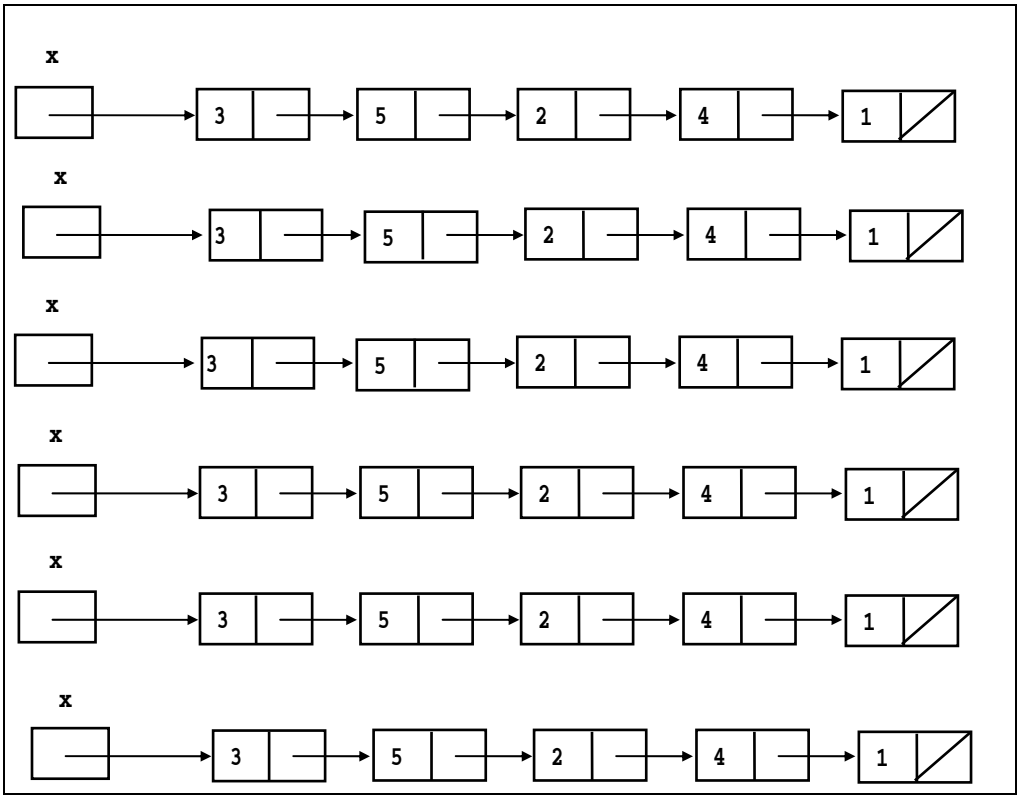
Recall the Academic Integrity statement that you signed. Write all answers clearly on these pages, ensuring your final answers are easily recognizable. The number of points for each problem is clearly marked, for a total of 25 points. I will post my solutions on the web on Monday, off the **Solutions** link, after my last class.

---

Throughout all the problems on this quiz, assume that we have declared the following list node class:

```
public class LN {
  public LN (int v, LN n) {value = v; next = n;}
  public int value;
  public LN  next;
}
```

1. (12 pts) Each linked lists is created from the class **LN**. Starting with each list fresh, indicate what state(s) change(s) when the statement to its left is executed. **Cross out** any values that are replaced and **Write in** new boxes or values (text or arrows).




In problems 2-4 on the next page, write first an **iterative** and then a **recursive** **static** method **countAdjacentDuplicates**, which each take one parameter: **l** a reference to a linked list (**LN** see above). This method counts and returns how many nodes contain values that are equal to the values in the node after them. The list **5**, **6**, **6**, **6**, **3**, **2**, **2**, and **6** returns **3** (first and second six are **1**; second and third six are **2**, first and second two are **3**). See the Weekly Schedule to download an Eclipse project folder to check your answers.

2. (4 points) Write **`countAdjacentDuplicates` iteratively**. Hint: Using a cursor **`r`**, be careful that comparing **`r.value`** and **`r.next.value`** never throws a **`NullPointerException`**. Of course, an empty list or list with one element has no adjacent duplicates.

```
public static int countAdjacentDuplicates (LN l) {
```

3. (4 points) Write **`countAdjacentDuplicates` recursively**. Many of the elements in the iterative solution will also appear in the recursive solution. Hint: My solution had two recursive calls in its body, with the code deciding which one to execute. Of course, an empty list or list with one element has no adjacent duplicates.

```
public static int countAdjacentDuplicates (LN l) {
```

4. (5 pts) Write a **`static`** method named **`copyWithSubstitution`**, which takes three parameters: a **`LN`** list and two **`int`** values The returned list is a **copy** of the first parameter (so, you **must** call the **`LN`** constructor; the first list must remain unchanged) with each occurrence of the first **`int`** replaced by the second **`int`**. **You must write this method recursively**. Hint: the code is a variant of the **`copy`** code we discussed for copying a linked list discussed in class.

```
public static LN copyWithSubstitution (LN list, int v1, int v2) {
```