This exam has 3 additional pages. Please answer each question on the page on which it is asked. You may write on the back of the facing page if you need to.
1. Class `SortedIntArrayList`, shown below, maintains `int` items in sorted order. The `SortedIntArrayList` can expand as needed. Most methods are not shown.

```java
class SortedIntArrayList {
    private int[] items;
    private int theSize;

    // Various methods omitted

    // If cap < theSize, throw an IllegalArgumentException exception
    // Otherwise, resize items so its length equals cap
    public void ensureCapacity(int cap) {
        /* You must implement */
    }

    // Remove item at specified index, maintaining sorted order
    // Throw ArrayIndexOutOfBoundsException exception if index is invalid.
    public void remove(int idx) {
        /* You must implement */
    }
}
```

(a) Implement `ensureCapacity`.
(b) Implement `remove`.
2. Assume that a `SimpleLinkedList` stores ints, with no duplicates. The list **IS NOT SORTED**.

Assume that the data representation of a `SimpleLinkedList` is as follows (observe the size is not maintained directly):

```java
private Node first; // the first node in the list; null if empty
private Node last;  // the last node in the list; null if empty
```

(a) Implement the `Node` class.
(b) Implement `removeFirst`. Be sure to correctly handle the special cases where the list has no elements and the list has one element.
(c) Implement `addLast`. Be sure to correctly handle the special case where the list is empty.
(d) Implement `size`. 
3. (a) Write an interface `cop3337.Multiset` with the public methods below. `Multiset` is the name of the generic interface that stores identically-typed items, allows duplicates, and has the following functionality:

- Four accessors: `count` returns the number of occurrences of a specified object (0 if it is not found at all), `isEmpty`, tests if the `Multiset` is empty; `size` returns the number of elements currently stored in the `Multiset` container, `uniqueSize` returns the number of unique elements currently stored in the `Multiset`.
  
  For instance, if the `Multiset` stores `[ 3, 4, 5, 3, 4 ]`, then `size` returns 5, but `uniqueSize` returns 3. `count(3)` returns 2, and `count(10)` returns 0.

- Two mutators: One makes the `Multiset` empty; the other (`add`) inserts a new item.

(b) Assume that a generic class `cop3337.TreeMultiset` implements the `Multiset` interface.

Implement static method `countUnique` that returns the number of unique items in its the array parameter. Implement `countUnique` by creating a `TreeMultiset` populating it with all the array items, and then invoking `uniqueSize`.

```java
// Return the number of unique strings in arr
// Create an appropriate multiset, add all items into it, and invoke
// multiset’s uniqueSize
public static int countUnique(String[] arr) {
```