CS 61B: Midterm Exam I

This is an open book, open notes exam. Electronic devices are forbidden on your person, including cell phones, iPods, headphones, and laptops. Turn your cell phone off and leave all electronics at the front of the room, or risk getting a zero on the exam. Do not open your exam until you are told to do so!

Name:_____________________________________
Login:____________________________________
Lab TA:____________________________________
Lab day and time:__________________________

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Possible</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Miscellaneous</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>2. Inheritance</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3. Run-length encoding</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

Do not write in these boxes.
Problem 1. (8 points) A miscellany.

a. (2 points) Explain why binary search on a doubly-linked list is no faster than searching the list from one end to the other.

b. (2 points) When the Java Virtual Machine throws a NullPointerException, how do you determine which part of your code threw the exception? How do you determine which operations in that code might have thrown the exception?

c. (2 points) Circle the lines of code that can never cause a run-time error in any context (not counting system errors such as running out of memory). Assume that all these lines compile without errors.

```java
boolean b1 = (head != null) && (head.next != null);
boolean b2 = (head == null) || (head.next == null);
Object o1 = (java.io.InputStream) System.in;
Object o2 = myObject.toString();
Comparable c = (Comparable) myObject;
int ia[] = {3, 7};
myStrings[1] = "Hello";
```

d. (1 point) Java has built-in arrays with a length field. But imagine if arrays were not built in to the language, and were instead somehow written out as a Java class definition. What modifiers would appear before length in its field declaration?

e. (1 point) Fill in the blanks: A cast changes the __________________________ type of an expression, and its effect on dynamic method lookup is __________________________.

Problem 2. (10 points) Inheritance.

On the next page, fill in the blanks so that the code compiles and runs without throwing an exception. Some blanks may require more than one word, or fewer. Make sure the class Rational and its subclasses Integ and WholeNum correctly implement rational numbers (fractions), integers, and whole numbers (nonzero integers), respectively. In particular, the method Rational.compareTo should return a negative value if this number is less than o, zero if the two numbers are equal, and a positive value otherwise, and it should correctly compare numbers among all three classes.

In the large box, override the compareTo method with an implementation that checks whether o is an Integ, calls the superclass compareTo if it is not, and otherwise does a faster comparison that does not look at the denom field.
public class Integr { // Invariant: denom == 1.
  public Integr(int n) {
    numer = n;
    denom = 1;
    if (n == 0) System.exit(1); // Enforce the invariant.
  }
  // Override compareTo to be fast if o is an Integr, call superclass otherwise.
}

public class WholeNum { // Invariants: denom == 1, numer != 0.
  public WholeNum(int n) {
    numer = n;
    denom = 1;
    if (n == 0) System.exit(1); // Enforce the invariant.
  }
  public short makesNoSense() {
    return Integr.lastCompareTo();
  }
}
Problem 3. (7 points) Run-length encoding an array of strings.

Write a method called rle in the SListNode class below that takes as input an array of Strings (the input parameter strings) and returns the head of a singly-linked list representing a run-length encoding of that array. Two adjacent Strings should be considered the same and compressed into a single run if they represent the same sequence of characters, even if they’re different objects.

There is no SList class; just an SListNode class. Each SListNode has three fields: an item reference that points to a String, an int named count that records the number of occurrences of that String, and a reference to the next SListNode in the list.

Feel free to manipulate references directly. Do not assume that any methods are available unless you write them, except the constructor provided below. Assume that strings references an array whose length is at least one, so the run-length encoding will have at least one node and rle never returns null. The run-length encoding is allowed to reference the same String objects as the array, but you can copy the Strings if you prefer. (Hint: the code is simpler if you start at the end of the array and encode backward.)

```java
public class SListNode {
    public String item;
    public int count;
    public SListNode next;

    public SListNode(String i, int c, SListNode n) {
        item = i; count = c; next = n;
    }

    public SListNode rle(String[] strings) { // Run-length encode an array.
        // Code implementation...
    }
}
```

Check here if your answer is continued on the back. [ ]