CS 455 Midterm Exam 2
Fall 2010 [Bono]
Nov. 10, 2010

There are 5 problems on the exam, with 55 points total available. There are 6 pages to the exam, including this one; make sure you have all of them. There is also a double-sided one-page code handout that accompanies the exam. If you need additional space to write any answers, you may use the backs of exam pages (just direct us to look there).

Remote DEN students only: Do not write on the backs of pages. If additional space is needed, ask proctor for additional blank page(s), put your name on them, and attach them to the exam.

Put your name and USC ID number at the top of the exam. Please read over the whole test before beginning. Good luck!

<table>
<thead>
<tr>
<th>Problem 1 &amp; 2</th>
<th>7 pts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 3</td>
<td>6 pts.</td>
</tr>
<tr>
<td>Problem 4A</td>
<td>15 pts.</td>
</tr>
<tr>
<td>Problem 4B</td>
<td>7 pts.</td>
</tr>
<tr>
<td>Problem 5</td>
<td>20 pts.</td>
</tr>
<tr>
<td>TOTAL</td>
<td>55 pts.</td>
</tr>
</tbody>
</table>
Problem 1 [2 pts.]
Part A. For each loop iteration of binary search in an array, we eliminate roughly how many elements from further consideration? Circle the one that matches most closely. In the following, \( k \) is the number of values that are still being considered at the start of that iteration.

a. 1  
b. \( k/2 \)  
c. \( \log_2 k \)  
d. \( k-1 \)

Part B. Answer the same question for linear search in an array. Circle the best choice:

a  b  c  d

Problem 2 [5 pts.]
Describe what the following method does in one or two sentences (i.e., something that might be appropriate in a method comment). You will get no credit for a line-by-line description of the code.

```java
// PRECONDITION: str.length() is even
public static boolean mystery(String str) {
    Stack<Character> stack = new Stack<Character>();
    int halfway = str.length() / 2;

    for (int i = 0; i < halfway; i++) {
        stack.push(str.charAt(i));
    }

    for (int i = halfway; i < str.length(); i++) {
        if (str.charAt(i) != stack.top()) {
            return false;
        }
        stack.pop();
    }
    return true;
}
```
Problem 3 [6 pts.]
Consider adding a new constructor to the *Poly* class that allows us to initialize a poly with \( n \) terms in one call. Its parameter is an *ArrayList* of *Terms*. Here is an example of its use to implement the *create* user command from PA2/3: (Note: the code here doesn’t deal with any user error processing).

```java
public static Poly doCreate(Scanner in) {
    ArrayList<Term> terms = new ArrayList();
    // getTerm gets the next term from the user. if there
    // were no more terms, returns null instead
    Term userTerm = getTerm(in);
    while (userTerm != null) {
        terms.add(userTerm);
        userTerm = getTerm();
    }
    return new Poly(terms);  // calls our new constructor
}
```

**Part A (4).** Describe in a few sentences how you would implement this new constructor (do not implement it). Assume we are using the same representation we did in PA3 (reminder about that and *create* command below).

**Part B (2).** Using big-O notation give the worst case run-time for calling the new constructor you described with an *ArrayList* of \( n \) terms.

---

**Reminder of *create* command:** Here is some example input, and the resulting polynomial:

```
$ cmd> create 1
Enter a space-separated sequence of coeff-power pairs terminated by <nl>
-3 10 3 1 0 1 52.3 0 5 2 12 0 3 10
$ cmd> print 1
5.0x^2 + 3.0x + 64.3
```

**Reminder of *Poly* representation:** Recall that the representation used by the *Poly* class is a Java *LinkedList* of non-zero terms in strictly decreasing order by exponent (strictly, meaning we only have at most one term that has a given exponent).
Problem 4 [22 pts. total]

Please refer to the code handout which shows a HashTable class that implements a hash table that maps from Strings to integers. It uses the auxiliary class, Entry, also on the code handout. Note: HashTable is not a Java class, and does not implement the exact Map interface.

Part A [15]. Implement the printAll method of HashTable (You are allowed to add private helper methods.)

    // prints all entries in the hash table, one entry per line
    // with a space separating the key from the value.
    // E.g., one line of output might be: joe 10
    public void printAll() {
Problem 4 (cont.)

Part B [7]. Consider a slightly different version of our HashTable class, that instead of providing the specific function, `printAll`, provides a similar, but more general function, `visitAll`, that takes a Java interface parameter of type `Visitor`. Here’s the header for the `visitAll` method and the complete `Visitor` interface:

```java
class HashTable {
    // visits each entry in the table, calling Visitor’s visit function
    // on each entry
    public void visitAll(Visitor visitor) {
    }
}

interface Visitor {
    public void visit(Entry entry);
}
```

Write the code necessary to use `visitAll` to add 10 points to all students’ scores in a HashTable of student names and scores called `namesAndScores`, defined below. You answer will involve completing the function below plus writing some additional code that will be necessary to make it work. Note: this problem is about using `visitAll`: do not implement `visitAll`.

```java
public static void add10toEach(HashTable namesAndScores) {
    namesAndScores.visitAll(
        // Code to add 10 points to each student's score
    );
}
```
Problem 5 [20 pts.]
Implement the static method `collapse` that returns a `LinkedList` like the one passed to it, but with all the runs “collapsed” to single elements. A run means two or more of the same value in a row.

Examples: (lists shown as space separated list of values surrounded by parentheses)

<table>
<thead>
<tr>
<th>list</th>
<th>return value of <code>compress(list)</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>(3 3 3 3)</td>
<td>(3)</td>
</tr>
<tr>
<td>(1 2 3)</td>
<td>(1 2 3)</td>
</tr>
<tr>
<td>()</td>
<td>()</td>
</tr>
<tr>
<td>(1 1 3 1 1 1)</td>
<td>(1 3 1)</td>
</tr>
<tr>
<td>(1 1 5 2 2 7 7 4)</td>
<td>(1 5 2 7 4)</td>
</tr>
<tr>
<td>(9)</td>
<td>(9)</td>
</tr>
</tbody>
</table>

// parameter list is unchanged by this method
public static LinkedList<Integer> collapse(LinkedList<Integer> list) {