CS314 Fall 2011 Midterm 1 Solution and Grading Criteria.

Grading acronyms:
AIOBE - Array Index out of Bounds Exception may occur
BOD - Benefit of the Doubt. Not certain code works, but, can't prove otherwise
ECF - Error carried forward.
Gacky or Gack - Code very hard to understand even though it works or solution is not elegant.
GCE - Gross Conceptual Error. Did not answer the question asked or showed fundamental misunderstanding
LE - Logic error in code.
NAP - No answer provided. No answer given on test
NN - Not necessary. Code is unneeded. Generally no points off
NPE - Null Pointer Exception may occur
OBOE - Off by one error. Calculation is off by one.

1. Answer as shown or -2 unless question allows partial credit.
No points off for differences in spacing, capitalization, commas, and braces
1 point for each part on G - J.

A. $T(N) = 5N^2 + 4N + 4$, +/- 1 on each coefficient
B. $O(N^2)$
C. O(N)
D. $O(N^2)$
E. $O(N^2)$ (worst case when all elements equal target)
F. $O(N^2)$ (worst case when no elements equal target)
G. 50 seconds
H. 44 seconds (if not simplified -2)
I. O(N)
J. 1. Invalid (can't instantiate interfaces)
   2. Valid
K. 1. Invalid (can't instantiate abstract classes
                and no default constructor)
   2. Invalid (POD is not a descendant of
                StorageUnit)
L. 50 1000
M. 1000 guard: true
N. 500 POD: 5
O. 0 [8, 16, 2] OR Syntax error (several people forgot the size of the initial list, which is 0. Differences in brackets and commas were ignored)

2. Comments. A relatively simple problem. Not algorithmically difficult. The real challenge was keeping track of the indices in each internal array and working with two different objects. A saw a lot of solutions different from the suggested solution that worked. I especially liked solutions that added a private method remove last.

Common problems:

- not tracking the index for the last element in each list. If the lists are different sizes the index of the last element in each list
- not reading the note (or remembering) that since the method is in the GenericList class you have access to all GenericLists' private instance vars, including other. Not necessary to write methods such as get(int x) and size().
- some solution relied on size staying constant, but then altered size in the loop.
- lots of off by one errors
- not stopping when one of the lists becomes empty
- using a nested loop to compare all pairs of elements

Suggested Solution:

```
public int trimEqualBacks(GenericList<E> other) {
    int indexThis = size - 1;
    int indexOther = other.size - 1;
    int result = 0;
    while(indexThis >= 0 && indexOther >= 0 &&
              values[indexThis].equals(other.values[indexOther])){
        result++;
        values[indexThis] = null;
        indexThis--;
        other.values[indexOther] = null;
        indexOther--;
    }

    size -= result;
    other.size -= result;

    return result;
 }
```

General Grading Criteria: 20 points

- track both indices correctly: 4
- loop while last elements equal: 2
- loop while still elements in both lists: 3
- use equals, not == to check equality of elements: 3
- null out elements removed from list: 2
- adjust sizes correctly: 3
- increment result for equal elements: 2
- return correct answer: 1

3. Comments: A very easy problem. Simple traversal of a 2d array.

Common problems:

- switching rows and columns
- off by one errors
- trying to call methods on the 2d array variable coefficients

Suggested Solution:

```
public int getNumUniformColumns() {
    int result = 0;
    for(int col = 0; col < coefficients[0].length; col++) {
        int first = coefficients[0][col];
        int row = 1;
        while(row < coefficients.length
                                && coefficients[row][col] == first)
            row++;
        if(row == coefficients.length)
            result++;
    }
    return result;
}
```

General Grading Criteria: 15 points

- outer loop for columns correct: 4 points
- move down rows in a column correctly: 4 points
- compare all coefficients correctly: 4 points (easiest way is compare to first)
- track number of uniform columns correctly: 2 points
- return result: 1 point

4. Comments. I thought this was the hardest problem on the test. Lot of abstractions going on. I was pleased people did so well on the question. I assume most people had never worked with maps before 314 and clearly most people have learned how use them and iterators correctly.

Common problems:

- not checking for names that were in the data map but not in the newRanks map and adding a 0 to those NameRecords
- not incrementing the numDecades instance variable
- hard coding the number of 0s. I am loathe to give examples on questions some times because then students hard code to the example. if numDecades was always 8 then it would have been a constant set to 8 and not a variable.
- not using iterator correctly

Suggested Solution

```
public void newDecade(Map<String, Integer> newRanks) {
    // create String with proper number of 0s
    String allZeros = " ";
    for(int i = 0; i < numDecades; i++)
        allZeros += "0 ";

// go through names in new ranks and update existing name records and add
// new NameRecords for names that have not appeared before
    for(String name : newRanks.keySet()) {
        NameRecord val = data.get(name);
        if(val == null)
            // new Record!
            data.put(name, new NameRecord(name + allZeros +
                                          newRanks.get(name)));
        else
            // already present, just update
            val.addRank(newRanks.get(name));
    }

    // now update all names that didn't appear in newRanks with a 0
    numDecades++;
    for(String name : data.keySet()) {
        NameRecord temp = data.get(name);
        if(temp.numDecades() != numDecades)
            temp.addRank(0);
    }
}
```

General Grading criteria: 20 points

- iterate through newRanks: 4
- check if name not present: 3
- create new NameRecord if not: 3
- correctly add appropriate number of 0s for new NameRecord: 3
- else update existing NameRecord: 3
- increment numDecades: 1
- update names present in data but not in newRanks: 3

5. Comments: Do you know how to move through a linked list? I saw a couple of nifty solutions using a for loop.

Common problems:

- not moving through list, temp = temp.getNext()
- destroying list
- off by one errors
- not using compareTo correctly or assuming return values are always -1, 0, or 1

Suggested Solution:

```
public int numLessThan(E tgt) {
    int result = 0;
    Node<E> temp = first;
    while(temp != null) {
        if(temp.getData().compareTo(tgt) < 0)
            result++;
        temp = temp.getNext();
    }
    return result;
}

// for loop version
public int numLessThan(E tgt) {
    int result = 0;
    for(Node<E> t = first; t != null; t = t.getNext())
        if(t.getData().compareTo(tgt) < 0)
            result++;
    return result;
}
```

General Grading criteria: 15 points

- assign temp to first: 2
- loop to end of list correctly: 3
- check data less than tgt correctly: 3
- increment result correctly: 1
- move temp down list correctly: 5
- return result: 1

IF

- destroy list -6
- use disallowed methods: -5
- off by one error (OBOE) -1
- not handling empty case correctly -2