Test 1 Practice: Compsci 201

Owen Astrachan

October 7, 2016

Name: ________________________________

NetID/Login: ____________

Honor code acknowledgment (signature) ________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>7 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>18 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>10 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 5</td>
<td>16 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 6</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>75 pts.</td>
<td></td>
</tr>
</tbody>
</table>

This test has 14 pages, be sure your test has them all. Do NOT spend too much time on one question — remember that this class lasts 75 minutes.

That means you should spend no more than 1 minute per point. Put your NetID clearly on each page of this test (worth 1 extra point).

In writing code you do not need to worry about specifying the proper import statements. Don’t worry about getting function or method names exactly right. Assume that all libraries and packages we’ve discussed are imported in any code you write. You can write any helper methods you would like in solving the problems. You should show your work on any analysis questions.

There are two blank pages at the end of the test. Make a note on the appropriate problem if you use the extra sheet.
PROBLEM 1: (LMNop (7 points))

For each of the methods below indicate the running time of the method in terms of the parameter $n$ (you can use $O$-notation or just provide an expression in terms of $n$.)

*Justify your answer for each or you will not receive credit.*

**Part A (2 points)**

Running time of call $\text{foo}(n)$ is:

```java
public double foo(int n){
    double sum = 1.0;
    for(int k=0; k < n; k++){
        sum += Math.pow(sum,k);
    }
    for(int k=0; k < n/2; k++){
        sum += 1;
    }
    return sum;
}
```

Running time of call $\text{bar}(n)$ is:

**Part B (2 points)**

```java
public int bar(int n){
    int sum = 0;
    for(int j=0; j < n; j += 1){
        for(int k=j; k >= 0; k -= 1){
            sum += n;
        }
    }
    return sum;
}
```

Running time of call $\text{frazz}(n)$ is:

**Part C (3 points)**

```java
public int frazz(int n){
    int p = 0;
    while (p*p < n*n){
        p += 1;
    }
    return p;
}
```
PROBLEM 2: \textit{(Complex or Imaginary (18 points))}

Consider the method \texttt{calc} below in the class \texttt{Test1}. When the \texttt{main} method is executed the output is

\begin{verbatim}
127
511
511
1023
\end{verbatim}

You’ll be asked several questions about this method and other methods below.

```java
public class Test1 {
    public int calc(int n) {
        int sum = 0;
        int val = 1;
        while (val <= n) {
            sum += val;
            val *= 2;
        }
        return sum;
    }

    public static void main(String[] args) {
        Test1 t = new Test1();
        System.out.printf("%d\n", t.calc(100));
        System.out.printf("%d\n", t.calc(256));
        System.out.printf("%d\n", t.calc(511));
        System.out.printf("%d\n", t.calc(512));
    }
}
```

\textbf{Part A (3 points)}
What is the value returned by each of the calls \texttt{calc(31)}, \texttt{calc(32)}, and \texttt{calc(33)}.

\textbf{Part B (3 points)}
What is the smallest value of \(x\) such that \texttt{calc(x)} returns 2047? Justify your answer briefly.
Part C (3 points)
In terms of $n$, what is the exact value returned by the call $\text{calc}(2^n)$, justify your answer briefly.

Part D (3 points)
What is the runtime complexity of the call $\text{calc}(n)$, use big-Oh and justify your answer.

Part E (4 points)
What is the value of the expression $\text{calc}(\text{calc}(256))$? In terms of $n$, what value is the expression $\text{calc}(\text{calc}(2^n))$ – base your answer to the latter question on what you wrote for Part C?
**Part F (6 points)**

Complete method `bigKey` that returns the key in parameter `map` whose associated value is the largest.

```java
public String bigKey(Map<String, Integer> map) {
    // Your implementation here
}
```
PROBLEM 3 :  *(Counting Boxes (13 points))*

The table below shows a five-rowed, two-dimensional structure that consists of rectangles. The rectangles are numbered starting from 1. After a row ends, the numbering continues to the next row. Each row has twice as many rectangles as the previous row.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
</tbody>
</table>

Please answer the questions below. Show your work to earn full credit.

For a and b, you may give an expression instead of an exact number for a and b (e.g., 7², log 7, etc.).

For c, d, and e, assume \( n \) is large.

A. [2pts] What is the exact value of the left-most number of the 7th row?

B. [2pts] What is the exact value of the right-most number of the 10th row?

C. [2pts] What is the big-Oh of the number of rectangles in the \( n \)th row? (include reason/justification)

D. [2pts] What is the big-Oh of the number of rows if the bottom row has \( n \) rectangles? (include reason/justification)

E. [2pts] What is the big-Oh of the number of rows if the whole structure has \( n \) rectangles? (include reason/justification)
PROBLEM 4: (Big Courses (12 points))

In this problem you'll write code to find information in data about enrollment in courses at Duke. The format of the data is as follows where each String represents one student (shown by first name) and the courses that student is taking — separated from the name by a colon ‘:’ and with each course separated by a comma ‘,’ from other courses.

String[] classes = {
    "owen:math216,compsci201,econ101,soc111",
    "fred:compsci201,econ101,evanth101,eos101",
    "mary:eos101,compsci92,music145,evanth101",
    "nancy:math216,phy141,chem201,compsci92",
    "fran:chem201,evanth101,soc111,psych101",
    "george:psych101,evanth101,chem201,pubpol165"
};

As an example of how to get information from this data, the method enrollment below returns the number of students enrolled in a specific course specified by parameter course. This output is generated by the code that follows using array classes as above.

2 compsci201
4 evanth101
2 eos101
3 chem201
0 art57

Here’s the code generating this output. You’ll be asked questions about this code and you’ll be asked to write more code manipulating this data.

import java.util.*;
public class ACES {

    public int enrollment(String[] list, String course){
        int count = 0;
        for(String s : list) {
            String[] first = s.split(":");
            String[] courses = first[1].split(",");
            if (Arrays.asList(courses).indexOf(course) >= 0){
                count++;
            }
        }
        return count;
    }

    public static void main(String[] args){
        String[] classes = /* not shown, see above */
        ACES m = new ACES();
        for(String s : new String[]{"compsci201","evanth101", "eos101", "chem201","art57"}){
            System.out.printf("%d\t%s\n",m.enrollment(classes, s),s);
        }
    }
}
Part A (4 points)
Briefly explain in words the purpose of both calls to `split` and the purpose of the call to `indexOf` in the implementation of `enrollment` that’s shown.

Part B (8 points)
Write the method `classMap` whose parameter is a list of student data and which returns a `Map<String,List<String>>` in which keys are course names and the value associated with each key is a list of students enrolled in the course. For example, for the key `compsci201` the value would be the list `"owen", "fred"` and for the key `chem201` the key would be the list `"nancy", "fran", "george"`.

You should return a `HashMap` in which the values are `ArrayList<String>` objects.

```java
import java.util.*;
public class ACES {

    public Map<String,List<String>> classMap(String[] list){
```

PROBLEM 5:  
*(Reasoning About Lists (16 points))*

The method `duplicate` in the code shown below changes parameter `list` so that it’s *doubled in place* – for example, the list

("ape", "bat", "cat", "dog")

is changed to the list below as a result of the call `duplicate(list)`.

("ape", "ape", "bat", "bat", "cat", "cat", "dog", "dog")

In this problem, the method `duplicate` is called with both an `ArrayList` and a `LinkedList` as parameters – the times for duplicating the different lists are shown tabularly and graphically below the code where the size of the list varies from 10,000 to 150,000 strings. The graphic is only for the `ArrayList` call to `duplicate` since the `LinkedList` timings are too small in comparison.

```java
public List<String> duplicate(List<String> list){
    ListIterator<String> iter = list.listIterator();
    while (iter.hasNext()){
        String s = iter.next();
        iter.add(s);
    }
    return list;
}
```

// code to call duplicate
for(int sz = 10000; sz <= 150000; sz += 10000){
    ArrayList<String> alist = createArrayList(sz);
    LinkedList<String> llist = createLinkedList(sz);
    // timing code not shown
    alist = duplicate(alist);
    llist = duplicate(llist);
    // print timings
}

<table>
<thead>
<tr>
<th>size (10^3)</th>
<th>link</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.001</td>
<td>0.056</td>
</tr>
<tr>
<td>20</td>
<td>0.001</td>
<td>0.168</td>
</tr>
<tr>
<td>30</td>
<td>0.002</td>
<td>0.381</td>
</tr>
<tr>
<td>40</td>
<td>0.002</td>
<td>0.681</td>
</tr>
<tr>
<td>50</td>
<td>0.003</td>
<td>1.063</td>
</tr>
<tr>
<td>60</td>
<td>0.003</td>
<td>1.567</td>
</tr>
<tr>
<td>70</td>
<td>0.005</td>
<td>2.129</td>
</tr>
<tr>
<td>80</td>
<td>0.005</td>
<td>2.780</td>
</tr>
<tr>
<td>90</td>
<td>0.006</td>
<td>3.509</td>
</tr>
<tr>
<td>100</td>
<td>0.007</td>
<td>4.458</td>
</tr>
<tr>
<td>110</td>
<td>0.008</td>
<td>5.33</td>
</tr>
<tr>
<td>120</td>
<td>0.009</td>
<td>6.408</td>
</tr>
<tr>
<td>130</td>
<td>0.010</td>
<td>7.467</td>
</tr>
<tr>
<td>140</td>
<td>0.010</td>
<td>8.401</td>
</tr>
<tr>
<td>150</td>
<td>0.010</td>
<td>9.654</td>
</tr>
</tbody>
</table>

\[ y = 0.0417x^2 + 0.0281x - 0.0723 \]
\[ R^2 = 0.9995 \]
Part A (4 points)
Using the same code on the same computer how much time will it take to duplicate both an `ArrayList` and a `LinkedList` list with 1,000,000 (one million) values. Justify your answers (your answer will be considered approximate, we’re looking for close enough.) *Don’t rely on the coefficients of the quadratic shown, use reasoning based on the empirical timings.*

Part B (4 points)
Using big-Oh, what is the complexity of duplicating both an N-element `ArrayList` and `LinkedList` using the code above. Justify your two answers empirically, by making explicit references to the timings.

Part C (4 points)
We haven’t discussed `ListIterator` in class, but the code runs quickly for `LinkedList` because new elements are added to the list, in place, as the iteration over the list takes place. Based on the code and your understanding what is the runtime of the single statement `iter.add(s)` in the code above is for both `LinkedList` and `ArrayList` lists, justify your answer.
Part D (4 points)

The code below is an alternate version of the `duplicate` code above. This code correctly modifies `list` so that it is doubled-in-place — its behavior is exactly the same as the code that uses the `ListIterator`, but the timings are different.

```java
public void duplicate2(List<String> list){
    int originalSize = list.size();
    list.addAll(list);

    for(int k=originalSize-1; k >= 0; k -= 1){
        String current = list.get(k);
        list.set(2*k, current);
        list.set(2*k+1, current);
    }
}
```

Explain why the `for` loop runs down to zero rather than up from zero (which would not work).

Provide a justification for why this code runs very quickly for `ArrayList` parameters and very slowly for `LinkedList` parameters. On the same computer on which the first timings were made, the `LinkedList` duplication takes 2.28 seconds for 20,000 strings and 18.7 for 40,000 strings whereas the `ArrayList` timings are all below 0.01 seconds.

Explain the timings by providing a reason for them based on the code and your understanding of how the classes `LinkedList` and `ArrayList` are implemented.
PROBLEM 6:  (Cookie’s Fortune (12 points))

For a Physics class with lab sections the professor uses a program that keeps attendance for each student. The information is stored in a map with the student name as the key and an array of labs that student has attended. These labs are stored as strings, e.g., the information below conveys the general idea of what’s stored in the map:

<table>
<thead>
<tr>
<th>key/name</th>
<th>labs attended</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Chris&quot;</td>
<td>[&quot;lab1&quot;, &quot;lab2&quot;, &quot;lab4&quot;, &quot;lab6&quot;]</td>
</tr>
<tr>
<td>&quot;Pat&quot;</td>
<td>[&quot;lab1&quot;, &quot;lab2&quot;, &quot;lab3&quot;, &quot;lab4&quot;]</td>
</tr>
<tr>
<td>&quot;Leslie&quot;</td>
<td>[&quot;lab1&quot;, &quot;lab3&quot;, &quot;lab5&quot;]</td>
</tr>
<tr>
<td>&quot;Gerry&quot;</td>
<td>[&quot;lab1&quot;, &quot;lab2&quot;, &quot;lab3&quot;, &quot;lab4&quot;, &quot;lab6&quot;]</td>
</tr>
</tbody>
</table>

The information is stored in a Java program with the type `HashMap<String, String[]>`. You should write a method that creates and returns a new map in which the key is a lab section, e.g., “lab1”, “lab3”, “lab5”, and the value associated with each key is an array of those students who attended that lab. For the information shown above all students attended lab1, but only three attended lab4. So part of the returned map is diagrammed below:

<table>
<thead>
<tr>
<th>key/lab</th>
<th>students in lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;lab1&quot;</td>
<td>[&quot;Chris&quot;, &quot;Pat&quot;, &quot;Leslie&quot;, &quot;Gerry&quot;]</td>
</tr>
<tr>
<td>&quot;lab2&quot;</td>
<td>[&quot;Chris&quot;, &quot;Pat&quot;, &quot;Gerry&quot;]</td>
</tr>
<tr>
<td>&quot;lab3&quot;</td>
<td>[&quot;Pat&quot;, &quot;Leslie&quot;, &quot;Gerry&quot;]</td>
</tr>
<tr>
<td>&quot;lab4&quot;</td>
<td>[&quot;Chris&quot;, &quot;Pat&quot;, &quot;Gerry&quot;]</td>
</tr>
<tr>
<td>&quot;lab5&quot;</td>
<td>[&quot;Leslie&quot;]</td>
</tr>
<tr>
<td>&quot;lab6&quot;</td>
<td>[&quot;Chris&quot;, &quot;Gerry&quot;]</td>
</tr>
</tbody>
</table>

You will write the method `makeLabList` that takes a map described in the first table above (where keys are student names) and returns a map in the second table (where keys are lab sections). First you’ll answer some questions about the method.

```java
/**
 * Returns a map that uses each string appearing in some value of the map
 * (representing lab sections) as keys with corresponding values
 * being an array of students who attended that lab
 */

public Map<String, String[]> makeLabList(Map<String, String[]> info) {

    Part A (2 points)
    Why does the parameter info have type `Map<String, String[]>` instead of `HashMap<String, String[]>` or `TreeMap<String, String[]>`.
```
**Part B (4 points)**

Writing the code that maps lab-sections to the students who attended the lab section requires using a structure like an ArrayList rather than an array to store the names of the students in a specific lab section as the code executes by looping over the map referenced by parameter `info`. Briefly, why will the map used in the method be defined as

```java
Map<String, ArrayList<String>> rev = new HashMap<String, ArrayList<String>>();
```

rather than as

```java
Map<String, String[]> rev = new HashMap<String, String[]>();
```

Please refer to the code on the next page to help understand the variable `rev` and how it is used.
public Map<String,String[]> reverse(Map<String,String[]> map) {
    Map<String, ArrayList<String>> rev = new HashMap<String, ArrayList<String>>();
    // add code here
    Map<String,String[]> ret = new HashMap<String,String[]>();
    for(String key : rev.keySet()) {
        ret.put(key, rev.get(key).toArray(new String[0]));
    }
    return ret;
}