Test 1: Compsci 201

Owen Astrachan

October 11, 2019

Name: ____________________________________________

NetID/Login: ________________

Community standard acknowledgment (signature) ______________________________

(1 point for name, netid)

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>8 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>12 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>34 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>75 pts.</td>
<td></td>
</tr>
</tbody>
</table>

This test has 11 pages, be sure your test has them all. Write your NetID clearly on each page of this test (worth 1 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.

You may consult your six (6) note sheets and no other resources. You may not use any computers, calculators, cell phones, or other human beings. Any note sheets must be turned in with your test.
PROBLEM 1:  (IO, IO, it's off to work (20 points))

Part A: 12 points
What is printed by each System.out.printf statement? There are four statements, each prints three values. Write the values below each printf statement.

String s = new String("DUKE");
String t = s;
String u = t.toLowerCase();

System.out.printf("%s 	 %s 	 %s\n", s, t, u);

String s = new String("DUKE");
String t = s;
String u = t.toLowerCase();

System.out.printf("%s 	 %s 	 %s\n", s, t, u);

String[] a = {"one", "two", "three", "four"};
String[] b = Arrays.copyOf(a,a.length);
String[] c = a;

System.out.printf("%s 	 %s 	 %s \
",a[0],b[0],c[0]);

String[] a = {"one", "two", "three", "four"};
String[] b = Arrays.copyOf(a,a.length);
String[] c = a;

System.out.printf("%s 	 %s 	 %s \
",a[0],b[0],c[0]);

String[] a = {"one", "two", "three", "four"};
String[] b = Arrays.copyOf(a,a.length);
String[] c = a;

System.out.printf("%s 	 %s 	 %s \
",a[0],b[0],c[0]);
Part B: 8 points
Write one of: True, False, or cannot be determined. Explain/Justify each answer

- If two string variables s and t have different hash code values, that is s.hashCode() != t.hashCode(), then what is the the value of s.equals(t)?

- If two string variables s and t have s.equals(t) is true; what is the the value of s == t)?

- If two string variables s and t have s.equals(t) is false; what is the value of s.hashCode() == t.hashCode().

- If two string variables s and t have s.equals(t) is true; and s.hashCode() == 7; what is the value of t.hashCode() == 7?
PROBLEM 2:  (I Sing Electric (8 points))

The code below is the implementing code from a version of the CelestialBody class that passes all tests for the method calcNetForceExertedByY.

```java
public double calcNetForceExertedByY(CelestialBody[] bodies) {
    double sum = 0.0;
    for(CelestialBody b : bodies) {
        if (b != this) {
            sum += calcForceExertedByY(b);
        }
    }
    return sum;
}
```

Part A (4 points)
The CelestialBody class has no overloaded operator for .equals() — if the expression (b != this) in the code above is replaced by (! b.equals(this)) will the code still pass all tests? Explain.

Part B (4 points)
The code below is from the simulation code in NBody.java and illustrates the call of method calcNetForceExertedByY shown above. Which of the three expressions from this calling code (line 98) corresponds to this in the implementing code (line 121 above):

* Explain your answer in few words.*

1. bodies[k]
2. bodies
3. yforces[k]

```java
double[] xforces = new double[ bodies.length ];
double[] yforces = new double[ bodies.length ];
for(int k=0; k < bodies.length; k++) {
    xforces[k] = bodies[k].calcNetForceExertedByX(bodies);
    yforces[k] = bodies[k].calcNetForceExertedByY(bodies);
}
```
As an example of how to think about some of the questions in this section, consider the method stuff below. The runtime complexity of this method is $O(n)$ and the value returned by the function is $O(n^2)$ for parameter $n$. As a concrete example, note than when $n = 100$ the loop executes 100 times doing an $O(1)$ operation each time. The value returned is $1 + 2 + \ldots + 99 = (98 \times 99)/2$. Note that even if the return statement was return sum*2 that the value returned would still be $O(n^2)$.

```java
public int stuff(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        sum += k;
    }
    return sum;
}
```

In all these problems $n$ is a positive number. In each problem you should provide two big-Oh expressions: one for runtime and one for value returned. Briefly justify each answer you provide. Your answers are for the entire method, justification can include discussing lines/loops in each method.

**Part A (4 points)**

What is the runtime complexity and the value returned by method evaluate below in terms of $n$? Use big-Oh and justify your answer briefly. Label the run-time and the value returned. Justify each answer.

```java
public int evaluate(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        for(int j=0; j < k; j++) {
            sum += 100;
        }
    }
    for(int k=0; k < n; k++){
        sum += 100;
    }
    return sum;
}
```
Part B (4 points)
What is the runtime complexity and the value returned by method calculate below in terms of \( n \)? Use big-Oh and justify your answer briefly. For this Part B the runtime and the value returned have the same big-Oh expression. If it helps, you can assume \( n \) is a power of 2

```java
public int calculate(int n){
    int sum = 0;
    for(int k=1; k < n; k *= 2){
        for(int j=0; j < n; j += 1){
            sum += 1;
        }
    }
    return sum;
}
```

Part C (4 points)
Consider the method stuff from the beginning of this problem, reproduced below. Recall that the runtime complexity is \( O(n) \) and the value returned is \( O(n^2) \).

```java
public int stuff(int n){
    int sum = 0;
    for(int k=0; k < n; k++){
        sum += k;
    }
    return sum;
}
```

Give big-Oh expressions for both the runtime complexity and the value returned for each of the expressions below. Justify your answers briefly.

```java
int x = stuff(stuff(n-10)); // big-Oh for runtime and value returned
int y = stuff(stuff(n*n)); // big-Oh for runtime and value returned
```
A multiset of strings can contain more than one occurrence of a string. The multiplicity of a string in a multiset is the number of times the string occurs. The size of a multiset is the number of different strings it contains. Consider this code example and the output that follows.

```java
Multiset s = new Multiset();
s.add("a");
s.add("a");
s.add("a");
s.add("b");
s.add("c");
s.add("c");
System.out.printf("%s\t%dn", s, s.size());

for(String ss : new String[] {"a","b","c","d","e"}){
    System.out.printf("%s\t%dn", ss, s.get(ss));
}
```

The output shows that in the Multiset s the String "a" has multiplicity three since it was added three times to the set and the set’s size is three since there are three unique strings: "a", "b", "c". Note that each element of the set is printed with its multiplicity so that "b" occurs once and "c" two times in the set. The code also shows how the Multiset.get method works to return the multiplicity of a string.

```
a:3,b:1,c:2 3
a  3
b  1
c  2
d  0
e  0
```

The last exam page contains Java code for a Multiset class with some methods missing that you’ll complete as part of this question.

**Part A (4 points)**

Methods toString and equals have an @Override annotation in the code, but methods size and add do not. Explain both why the error message does not override method in superclass is generated if @Override is added before size, and not generated when added before equals.
Part B (6 points)

A multiset a contains another multiset b if every element in b occurs in a with the same or greater multiplicity. In other words, if some element in b occurs more often in b than in a, then a does not contain b. The code below shows examples of how contains is called – the output from this code follows. When the code runs, Multiset a already has values as seen when it is printed.

```java
System.out.println(a);
Multiset d = new Multiset(a);
System.out.println(a.equals(d));
System.out.printf("%s contains %s: %s\n",a,d,a.contains(d));
a.add("a");
System.out.printf("%s contains %s: %s\n",a,d,a.contains(d));
d.add("a"); d.add("a");
System.out.printf("%s contains %s: %s\n",a,d,a.contains(d));
```

a:2,b:3,c:1
true
a:2,b:3,c:1 contains a:2,b:3,c:1: true
a:3,b:3,c:1 contains a:2,b:3,c:1: true
a:3,b:3,c:1 contains a:4,b:3,c:1: false

Complete the implementation of the method Multiset.contains.

```java
public boolean contains(Multiset a) {
...
```
Part C (4 points)
The Multiset class has two constructors. Code shown earlier/above in this problem illustrates both constructors being called. Briefly, explain why the code below will always print true, regardless of how many times a.add(...) is called. You must refer to the constructor used in creating Multiset b and to the .equals method in your response.

Multiset a = new Multiset();
// call a.add(..) multiple times
Multiset b = new Multiset(a);
System.out.println(a.equals(b));

Part D (4 points)
The cardinality of a Multiset is the total of all multiplicities of strings in the multiset (the total number of strings including duplicates since a multiset can contain multiple occurrences of a string). The code below and its output show two calls of the method cardinality for two different multisets. Note that the cardinality is the sum of the multiplicities for each set.

```
35 System.out.printf("%s --- %d\n",b,b.cardinality());
36 System.out.printf("%s --- %d\n",a,a.cardinality());
```

a:1,b:2,d:4 --- 7
a:2,b:3,c:1 --- 6

If a.contains(b) is true, is b.cardinality() <= a.cardinality() always true? Explain.

If a.contains(b) is false, is b.cardinality() <= a.cardinality() always false? Explain.
**Part E (6 points)**
Implement the method `cardinality`

```java
    public long cardinality() {
        long sum = 0;
        return sum;
    }
```

**Part F (2 points)**
The instance variable `myMap` stores integer values as the multiplicity of each string in a `Multiset`, but the return type of `cardinality` is `long`. Briefly explain why `long` is useful as the return type even though integer values are stored.
Part G (4 points)
Currently, client code must call the method .add("duke") ten times to add ten occurrences of "duke" to a multiset.

Multiset a = new Multiset();
for(int k=0; k < 10; k++) a.add("duke");

Suppose a new, overloaded method add method is included in the Multiset class so that the call a.add("duke",10) works to add "duke" ten times. Complete the implementation below

public void add(String key, int occurrences) {

}