

Test 1 : Compsci 201

Owen Astrachan

February 15, 2019

Name: _____

NetID/Login: _____

Community standard acknowledgment (signature) _____

	value	grade
Problem 1	22 pts.	
Problem 2	14 pts.	
Problem 3	9 pts.	
Problem 4	18 pts.	
Problem 5	10 pts.	
TOTAL:	73 pts.	

This test has 12 pages, be sure your test has them all. Write your NetID *clearly* on each page of this test and on the front of this test (worth 2 points).

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 : (*Zeus and a Moon of Jupiter (22 points)*)**Part A: 8 points**

What is printed by each `System.out.printf` statement? There are four statements, each prints two things.

```
String s = new String("duke");
String t = s;

String[] a = {"bat", "dog", "cat", "ant", "fox"}
String[] b = a;

s = s + "duke";

a[1] = a[4];

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

System.out.printf("%s \t %s\n",a[1], b[1]);
System.out.printf("%s \t %s\n",b[4]+a[4], a[4] + b[4]);
```

Part B: 6 points

The code below prints 3.

```
23     ArrayList<String> a = new ArrayList<>();
24     String[] array = {"red", "orange", "yellow"};
25     a.addAll(Arrays.asList(array));
26     System.out.println(a.size());
```

Two integers are printed by the code below, which runs immediately after the code above. The first integer printed is 12. Explain why. What is the second integer printed? Explain your answer.

```
28     for(int k=0; k < 3; k++) a.addAll(Arrays.asList(array));
29     System.out.println(a.size());
30     for(int k=0; k < 3; k++) a.addAll(a);
31     System.out.println(a.size());
```

Part C: 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 == t` is true; what is the the value of `s.equals(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; what is the value of `s == t`.

PROBLEM 2 : (Array Growth (14 points))

In the `GrowableStringArray` class used in the *diyad array* assignment two methods were part of adding values to the growable array:

```

13 public void add(String s) {
14     checkSize();
15     myStorage[mySize] = s;
16     mySize++;
17 }
18
19 private void checkSize() {
20     if (mySize >= myStorage.length) {
21         String[] storage = new String[(int) (myStorage.length * 2)];
22         System.arraycopy(myStorage, 0, storage, 0, myStorage.length);
23         myStorage = storage;
24     }
25 }

```

Two runs are shown below, the size of the array created by adding one element at a time is in the left-most column; the timings from the code shown above are in the middle column; and timings from when `* 2` is replaced by `+ 100` are shown in the right column.

array size	* 2	+ 100
100000	0.004	0.093
200000	0.005	0.570
300000	0.005	1.243
400000	0.009	2.203
500000	0.006	3.504
600000	0.008	4.878
700000	0.007	6.970
800000	0.008	9.141
900000	0.009	10.224
1000000	0.013	13.342

Part A (3 points)

Approximately (within ± 3) how many times are lines 21-23 executed if the initial size of `myStorage` is one, and a total of 1,025 elements are added to the array by calling method `add` shown above 1,025 times and where `checkSize` uses `* 2` when the array grows. *Justify briefly.*

Part B (4 points)

The data generating the column on the right uses + 100. If + 100 is replaced by + 1000 will the timings in the last column be less or more than what's shown for + 100. Explain your answer, be brief.

Part C (4 points)

The timings in the middle column use * 2. If * 1.25 is used instead will the timings be more similar to the middle column or more similar to the right column? Explain your answer.

Part D (3 points)

In the code for `checkSize` new array storage is created on line 21. Assume a total of N values are added when `myStorage` has room initially for one `String` and * 2 is used to grow the array, as shown.

Using O -notation, what is the total amount of storage created? Justify your answer. If it helps your reasoning, you may assume that $N = 2^k$ for some k , i.e., that N is a power-of-two. Recall that in class we noted that $1 + 2 + 4 + 8 + \dots + 2^k = 2^{k+1} - 1$

PROBLEM 3 : (*I ain't got NBody (9 points)*)

The code below is from a version of the `CelestialBody` class that passes all tests.

```
104 public double calcNetForceExertedByX(CelestialBody[] bodies) {
105     double s = 0.0;
106     for(CelestialBody b : bodies) {
107         if (! b.equals(this)) {
108             s += calcForceExertedByX(b);
109         }
110     }
111     return s;
112 }
```

Part A (3 points)

If the expression `(! b.equals(this))` is replaced by `(! this.equals(b))` will the code still pass all tests? Explain.

Part B (3 points)

The code calls `.equals` and works correctly for the `NBody` simulation even though `.equals` is **not overridden** in the `CelestialBody` class.

Explain why the use of `.equals` works in the simulation even though there is no overridden method.

Part C (3 points)

What type of value is returned by the method `calcForceExertedByX` ?

PROBLEM 4 : (*Sloths, Tenors, Musketeers (18 points)*)

The code for a class `Triangle` is given after the next page as part of this problem – with a complete class definition and a `main` driver. *Be sure to review the code.*

The output of running the program is:

```
[3, 4, 5] 12 6.00
[8, 15, 17] 40 60.0
[10, 10, 12] 32 48.0
```

Part A (2 points)

Explain why 12, 40, and 32 are printed as shown above.

Part B (4 points)

If the statement `new Triangle(10,10,12)` in `main` is replaced by `new Triangle(10,10,20)` the program does **not** print anything other than an error *IllegalArgumentException: bad sides 10 10 20*.

What code generates this exception? (reference a line number).

A valid triangle in geometry must have the sum of two sides greater than the maximal/largest side. Explain why testing only `mySideA + mySideB < mySideC` is sufficient and other orderings of sides aren't needed. Reference code.

Part C (4 points)

What is the purpose of the `@Override` annotation for `toString`. Why isn't `@Override` used before the method `perimeter`?

Part D (4 points)

Assume the method `trianglesFromFile` is uncommented and called from `main`, and the file `triangles.txt` has these six lines:

```
3 4 5
3 4 5
3 4 5
3 4 5
3 4 5
3 4 5
```

The output generated by the call `trianglesFromFile` is as shown. Although all six triangles are the same, the set contains six elements. Explain conceptually why the set contains six `Triangle` objects instead of one triangle object. Reference the method `public boolean equals(Object o)` used by the `Triangle` class. In the next question you'll rewrite this method.

```
# triangles = 6
[3, 4, 5]
[3, 4, 5]
[3, 4, 5]
[3, 4, 5]
[3, 4, 5]
[3, 4, 5]
```


02/13/19
10:12:27

Triangle.java

1

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class Triangle {
    private int mySideA;
    private int mySideB;
    private int mySideC;

    public Triangle(int a, int b, int c) {
        int[] sides = {a,b,c};
        Arrays.sort(sides);
        mySideA = sides[0];
        mySideB = sides[1];
        mySideC = sides[2];

        if (mySideA + mySideB <= mySideC) {
            throw new IllegalArgumentException(String.format("bad sides %d %d %d", a,b,c));
        }
    }

    public int perimeter() {
        return mySideA + mySideB + mySideC;
    }

    public double area() {
        double p = perimeter()/2.0;
        return Math.sqrt(p*(p-mySideA)*(p-mySideB)*(p-mySideC));
    }

    @Override
    public int hashCode() {
        return perimeter()*mySideC;
    }

    @Override
    public String toString() {
        return String.format("[%d, %d, %d]",mySideA, mySideB, mySideC);
    }

    public static void main(String[] args) throws FileNotFoundException {
        Triangle[] triangles = {
            new Triangle(3,4,5),
            new Triangle(8,15,17),
            new Triangle(10,10,20),
        };
        for(Triangle t : triangles) {
            System.out.printf("%s\t%d\t%2.3g\n", t, t.perimeter(), t.area());
        }

        //trianglesFromFile();
    }

    public static void trianglesFromFile() throws FileNotFoundException {
        HashSet<Triangle> set = new HashSet<>();
        Scanner scan = new Scanner(new File("data/triangles.txt"));
        while (scan.hasNextInt()){
            int a = scan.nextInt();
            int b = scan.nextInt();
            int c = scan.nextInt();
            Triangle t = new Triangle(a,b,c);
            set.add(t);
        }
    }
}
```

```
System.out.printf("# triangles = %d\n",set.size());
for(Triangle t : set) {
    System.out.println(t);
}
scan.close();
}
```

Part E (4 points)

Complete method `equals` so it works as intended. If you implement it correctly the code in `trianglesFromFile` should generate the output below after reading the same data file. The code you write should ensure that two triangles with sides `[3,4,5]` are equal, but that triangles with sides `[3,4,5]` and `[4,4,4]` are *not equal*. Two triangles are equal when they have three equal sides.

```
# triangles = 1
[3, 4, 5]
```

Complete the method below.

```
public boolean equals(Object o) {
    if (o == null || !(o instanceof Triangle)) return false;

    Triangle t = (Triangle) o;
```

```
}
```

PROBLEM 5 : (*Amp, Pam, Map (10 points)*)

The `main` method shown on the next page is missing statements in the while loop body that will generate the output below. This shows there are 49 two-letter words; 2,235 four-letter words; and 103 16-letter words in the file `wordslower.txt` – a text file of over 45,000 words in the English language.

output of running code

```

2      49
3      535
4     2235
5     4170
6     6166
7     7359
8     7070
9     6079
10    4591
11    3069
12    1880
13    1137
14    545
15    278
16    103
17     57
18     23
19     3
20     3
21     2
22     1
28     1

```

explanation of code shown on next page

The code uses a map in which keys are integer word lengths and the corresponding value for each key is a list of words with that word length.

For example, the statement

```
System.out.println(map.get(19))
```

prints three strings in an `ArrayList` as shown:

```
[anthropomorphically, incomprehensibility, straightforwardness]
```

Part A (2 points)

Assuming that `wordslower.txt` contains most of the words in the English language, what are two possible words of the 49 words printed by `System.out.println(map.get(2))`

Part B (2 points)

Why is there output for lines labelled 20, 21, and 22, but then no output for 23-27? Reference the contents of the file `wordslower.txt` in your answer.

Part C (6 points)

Complete the `while` loop so that the code works and generates the output shown above. It's possible to complete the code with one call of `map.putIfAbsent` and one call of `map.get` with the right parameters, but you can write as many and any lines that result in correct output.

```
public static void main(String[] args) throws FileNotFoundException {
    File wfile = new File("/data/wordslower.txt");
    Scanner scan = new Scanner(wfile);

    Map<Integer,ArrayList<String>> map = new HashMap<>();

    while (scan.hasNext()) {
        String s = scan.next();
        int len = s.length();
        // add code here

    }

    for(int len : map.keySet()) {
        System.out.printf("%d \t%6d\n",len,map.get(len).size());
    }
    scan.close();
}
```