Test 1 Fall 2018 : Compsci 201

Owen Astrachan

September 26, 2018

Name: _____

NetID/Login: _____ (1 point)

Community standard
acknowledgment (signature) _____ (1 point)

	value	grade
Problem 1	28 pts.	
Problem 2	9 pts.	
Problem 3	16 pts.	
Problem 4	20 pts.	
TOTAL:	75 pts.	

This test has 16 pages, be sure your test has them all. Please write your NetID *clearly* on each page of this test. You will get one point for including your netid on the first page. There is one blank page at the end of the test.

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 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 : (*I-O, I-O, it's off to the pasture we go (12 points)*)

Part A: 10 points

What is printed by each `System.out.printf` statement? Indicate the output next to each statement. There are five statements.

```
String s1 = new String("crazy");
String s2 = s1;
String s3 = s2 + s2;

String[] a = {"red", "orange", "yellow", "green"}
String[] b = a;

s1 = s1 + " love";

b[3] = "violet";

System.out.printf("%s \t %d\n",s1, s1.length());
System.out.printf("%s \t %d\n",s2, s2.length());
System.out.printf("%s \t %d\n",s3, s3.length());

System.out.printf("%s %s\n",a[3], b[3]);
System.out.printf("%s\n", a[0] + b[0]);
```

Part B: 6 points

True or False, Explain/Justify each answer for full credit.

- Assume that for two string variables `s` and `t` the value of `s == t` is *true*. In this case the value of `s.equals(t)` *must* also be *true*.

- Assume that for two string variables `s` and `t` the value of `s == t` is *true*. In this case `s.hashCode() == t.hashCode()` *must* also be *true*.

- Assume that for two string variables `s` and `t` the value of `s == t` is *false*. In this case `s.hashCode() == t.hashCode()` *must* also be *false*

Parts C and D: 12 points

Questions in this section are based on the class `Point` shown below. You were shown this class for an assignment quiz.

```
public class Point {
    private double myX;
    private double myY;

    public Point(double x, double y) {
        myX = x;
        myY = y;
    }

    @Override
    public String toString() {
        return String.format("(%2.3f,%2.3f)", myX, myY);
    }

    /**
     * return Euclidean distance from this point to another point
     * @param other Point to which distance calculated
     * @return distance from this point to other
     */
    public double distanceFrom(Point other) {
        double dx = myX - other.myX;
        double dy = myY - other.myY;
        return Math.sqrt(dx*dx + dy*dy);
    }

    @Override
    public boolean equals(Object o) {

        Point other = (Point) o;
        if (other.myX == myX && other.myY == myY) {
            return true;
        }
        return false;
    }

    @Override
    public int hashCode() {
        return (int) (100*myX + myY);
    }
}
```

Part C: 6 points

The output of each `println` statement below is either *true* or *false*. Indicate the value and provide a brief reason for your answer.

```
 Point a = new Point(13,77);
 Point b = new Point(92,158);

System.out.println(a.distanceFrom(b) == b.distanceFrom(a));




System.out.println(a.hashCode() == b.hashCode());




System.out.println(a.equals(b));
```

Part D: 6 points

The code below reads a file containing an even number of `double` values and should print the point furthest from the origin. For the data file shown below

```
5 12 3 4 -3 -4 4 -3
6 8 -6 8 -6 -8 6 -8
```

The output should be

```
furthest point is (5.000,12.000);
distance = 13.000
```

Complete the code so it works as intended. You should add code within the `while` loop in two places as indicated in the code.

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

public class PointDriver {
    public static void main(String[] args) throws FileNotFoundException {
        Scanner scan = new Scanner(new File("points.txt"));
        Point origin = new Point(0,0);
        Point max = origin;

        while (scan.hasNextDouble()) {

            // make two scan.nextDouble calls to read two double values









            // add code here to create a Point object and to update max

















        }
        System.out.printf("furthest point is %s\n", max);
        System.out.printf("distance = %1.3f\n",max.distanceFrom(origin));
        scan.close();
    }
}
```

PROBLEM 2 : (*I ain't got nobody (9 points)*)

Code for a partially complete, but correct `Body.java` file from the *NBody* assignment is shown on the next page. Use this code and your knowledge of the assignment to answer these questions.

Part A (3 points)

Can one of the `Body` constructors be written with a single line using `this` and calling the other constructor? Justify/explain your answer briefly.

Part B (3 points)

In the method `calcDistance` the expression `myXPos - b.myXPos` is used. Can the expression be replaced with `getX() - b.getX()` without affecting the correctness of the method? Justify/explain your answer.

Part C (3 points)

The code shown in `calcNetForceExertedByX` references `this` in the boolean expression `! b.equals(this)` Explain in words the purpose of the if statement in this method making explicit reference to the parameter `bodies`.

```
/**
 * Celestial Body class for NBody
 * @author ola
 *
 */
public class Body {
        private double myXPos;
        private double myYPos;
        private double myXVel;
        private double myYVel;
        private double myMass;
        private String myFileName;

        /**
         * Create a Body from parameters
         * @param xp initial x position
         * @param yp initial y position
         * @param xv initial x velocity
         * @param yv initial y velocity
         * @param mass of object
         * @param filename of image for object animation
         */
        public Body(double xp, double yp, double xv,
                    double yv, double mass, String filename){
            // code not shown
        }

        /**
         * Copy constructor: copy instance variables from one
         * body to this body
         * @param b used to initialize this body
         */
        public Body(Body b){
            // code not shown
        }

        public double getX() {
            return myXPos;
        }
        public double getY() {
            return myYPos;
        }
        public double getXVel() {
            return myXVel;
        }
        /**
         * Return y-velocity of this Body.
         * @return value of y-velocity.
         */
        public double getYVel() {
            return myYVel;
        }

        public double getMass() {
            return myMass;
        }
        public String getName() {
            return myFileName;
        }

        /**
         * Return the distance between this body and another
         * @param b the other body to which distance is calculated
         * @return distance between this body and b
         */
        public double calcDistance(Body b) {
            double dx = myXPos - b.myXPos;
            double dy = myYPos - b.myYPos;
            return Math.sqrt(dx*dx + dy*dy);
        }

        public double calcForceExertedBy(Body p) {
            // not shown
        }

        public double calcForceExertedByX(Body p) {
            // not shown
        }
        public double calcForceExertedByY(Body p) {
            // not shown
        }

        public double calcNetForceExertedByX(Body[] bodies) {
            double s = 0.0;
            for(Body b : bodies) {
                if (! b.equals(this)) {
                    s += calcForceExertedByX(b);
                }
            }
            return s;
        }

        public double calcNetForceExertedByY(Body[] bodies) {
            // not shown
        }

        public void update(double deltaT,
                           double xforce, double yforce) {
            // not shown
        }

        public void draw() {
            // not shown
        }
}
```

PROBLEM 3 : (*What does the Hen Weigh, What does the Fox Say (16 points)*)

In this question you'll reason about the method `countLess` shown below. Three calls and the return values for these calls are given before the code. Parameters `a` and `b` are strings that are each a space-delimited sequence of "words" as seen in the examples.

Call	Return	Reason
`countLess("fox hen fox", "hen hen fox fox")`	*false*	fox occurs the same number of times
`countLess("fox hen fox", "hen hen fox fox fox")`	*true*	fox and hen both less
`countLess("b b", "b b b")`	*true*	two b's and three b's

```
public boolean countLess(String a, String b) {
    List<String> alist = Arrays.asList(a.split(" "));
    List<String> blist = Arrays.asList(b.split(" "));

    for(String s : alist) {
        int ac = Collections.frequency(alist, s);
        int bc = Collections.frequency(blist,s);
        if (ac >= bc) return false;
    }
    return true;
}
```

Part A: 4 points

What do the calls `a.split(" ")` and `(b.split(" ")` return? Provide two answers: the Java type these methods return and their conceptual use in the code.

Part B: 4 points

Explain briefly the purpose of creating the variables `alist` and `blist` in the code above. Reference their use in the calls to `Collections.frequency`

Part C: 4 points

Provide an example of Strings `a` and `b` such that the call `countLess(a,b)` returns `true`, `a` contains six "words", `b` contains seven "words", and `Collections.frequency` is called a total of 12 times. Explain/Justify your answer.

Part D: 4 points

Provide an example of Strings `a` and `b` such that the call `countLess(a,b)` returns `false`, `a` contains six "words", `b` contains seven "words", and `Collections.frequency` is called a total of two times. Explain/Justify your answer.

PROBLEM 4 : (*Solitary Royalist (20 points)*)

Two words are *anagrams* if the letters in one word can re-arranged to obtain the other word. For example: *emigrants*, *mastering*, and *streaming* are anagrams of each other as are *leap*, *pale*, *peal*, and *plea*.

The use of class `Anagram` is shown below. The output of this code appears first and illustrates how words that are anagrams of each other compare as equal.

Output first:

cone once true

dog cat false

act cat true

Code generating output

```
Anagram a = new Anagram("cone");
Anagram b = new Anagram("once");
System.out.printf("%s %s %s\n",a,b,a.equals(b));

Anagram c = new Anagram("dog");
Anagram d = new Anagram("cat");
System.out.printf("%s %s %s\n", c,d,c.equals(d));

c = new Anagram("act");
System.out.printf("%s %s %s\n", c,d,c.equals(d));
```

The code for the `Anagram` class is given after this problem (the last non-blank page on this test).

The `Anagram` class uses a concept similar to the idea seen in the APT *Anonymous* – an `Anagram` is constructed from a String and counts of how many occurrences of each letter 'a' through 'z' are kept in an array. These counts determine if two `Anagram` objects are equal. You can see this in the code for the class shown on the last exam page. The diagram below shows the array of counts for the `Anagram` object created by `new Anagram("impressive")` – note that the number of occurrences of `'e'` is stored in `myCounts[4]`. These same counts would be stored for the object created by `new Anagram("permissive")` since these are anagrams of each other.

0	0	0	0	2	0	0	0	2	0	0	0	1	0	0	1	0	1	2	0	0	1	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

(questions on next page)

Part A (4 points)

What is printed by this code (two lines are printed, write your answer below each print statement).

```
Anagram a = new Anagram("rat");
Anagram b = new Anagram("star");
Anagram c = b;

System.out.printf("%s %s %s\n",a,b,a.equals(b));




System.out.printf("%s %s %s\n",b,c,b.equals(c));
```

Part B (3 points)

In the code at the beginning of the problem the first line of output shown is `cone once true`. Explain why the `Anagram` objects print differently, i.e., as cone and once, but the objects are still equal. Make explicit reference to methods in the `Anagram` code.

Part C (3 points)

The code below in method `anaStats` reads a file containing 45,356 different English words.

The code prints `list = 45356, set = 42234`

Explain why the set has fewer elements than the list. Be brief. Your answer should be conceptual. You do not need to make reference to specifics of the `Anagram` implementation.

```
public void anaStats() {
    File wfile = new File("/data/wordslower.txt");
    Scanner scan = new Scanner(wfile);
    ArrayList<Anagram> list = new ArrayList<>();
    HashSet<Anagram> set = new HashSet<>();
    while (scan.hasNext()) {
        String s = scan.next();
        Anagram ana = new Anagram(s);
        list.add(ana);
        set.add(ana);
    }
    System.out.printf("list = %d, set = %d\n",
                      list.size(), set.size());
}
```

Part D (4 points)

The code in `anaStats` on the previous page takes 0.868 seconds to run on a reasonably powered laptop.

If the return statement in `Anagram.hashCode` is changed to `return myCounts[0] + myCounts[3]` the code takes 16.591 seconds to run, but prints the same values. If the return statement is changed to `return 5` the code takes 59.437 seconds to run, but still prints the same values.

Briefly explain why the same, correct values are printed each time, and why the runtime increases for each example. Be brief. You should be sure to address the increase in runtime and why one increase is greater than the other.

Part E (6 points)

The code shown on the next page prints these five lines of output.

```
caret cater crate react recta trace
pares parse pears rapes reaps spare spear
caster caters crates reacts recast traces
opts post pots spot stop tops
arrest rarest raster raters sartre starer
```

```
public static void main(String[] args) throws FileNotFoundException {
    File wfile = new File("/data/wordslower.txt");
    Scanner scan = new Scanner(wfile);
    Map<Anagram,List<String>> map = new HashMap<>();

    while (scan.hasNext()) {
        String s = scan.next();
        Anagram ana = new Anagram(s);
        if (! map.containsKey(ana)) {
            map.put(ana, new ArrayList<String>());
        }
        map.get(ana).add(s);
    }

    for(Anagram ana : map.keySet()) {
        if (map.get(ana).size() > 5) {
            for(String s : map.get(ana)) {
                System.out.print(s+" ");
            }
            System.out.println();
        }
    }
```

In words, explain the output shown and why only some keys and values of the map are printed. Specifically you must answer three questions:

Why does each line contain words that are anagrams of each other. Be sure to reference how the map is used and what keys and values are in the map.

Explain why the line `introduces reductions` is not printed as well as in general what keys/values in the map are **not** printed.

Explain how the output will change if the 5 is changed to a 6 in the code.

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

public class Anagram {
	private int[] myCounts;
	private String myString;

	public Anagram(String s) {
		myString = s;
		count(s.toLowerCase());
	}

	private void count(String s) {
		myCounts = new int[26];
		for(char ch : s.toCharArray()) {
			int index = ch - 'a';
			if (index < 0) continue;
			myCounts[index] += 1;
		}
	}

	@Override
	public String toString() {
		return myString;
	}

	@Override
	public boolean equals(Object o) {
		if (this.getClass() != o.getClass()) {
			return false;
		}

		Anagram ana = (Anagram) o;
		for(int k=0; k < myCounts.length; k += 1) {
			if (myCounts[k] != ana.myCounts[k]) {
				return false;
			}
		}
		return true;
	}

	@Override
	public int hashCode() {
		int sum = 0;
		for(int k=0; k < myCounts.length; k+=1) {
			sum += (k+1)*myCounts[k];
		}
		return sum;
	}
}
```