CS 201, Fall 2013 — Dec 13th — Final Exam Name: Solution

Note: In questions where you are asked about a static method, assume that the method is in a class called Qn where n is the question number, e.g., Q1 for Question 1.

Question 1. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

a and b are different variables – assigning to a doesn't affect b

```
public class Q1 {
  public static void f(int[] a) {
    a = new int[1];
    a[0] = 42;
  }
```

```
  public static void main(
      String[] args) {
    int[] b = new int[1];

    b[0] = 17;
    f(b);
    System.out.printf("%d\n", b[0]);
  }
}
```

17

Question 2. [5 points] Consider the following class and static method:

```
public class Animal {
  public String sound() {
    return "unknown";
  }
}
```

```
public class Q2 {
  public static void mystery(Animal a) {
    System.out.println(a.sound());
  }
}
```

Is it possible to predict with certainty what output will be printed by the mystery method when it is called? If so, what is the output? If not, why not? Explain briefly.

Hint: Notice that the code does not contain any specific call to the mystery method.

No, because the parameter a could refer to an instance of a subclass of Animal whose sound() method returned an arbitrary String

Question 3. [10 points] Specify code that can be used at the points labeled `Missing code 1` and `Missing code 2` to allow the following program to compile and, when the `main` method in the `ColorItem` is executed, produce the output `Apples,3,red`.

```java
public class Item {
  private String name;
  private int quantity;

  public Item(String n, int q) {
    name = n;
    quantity = q;
  }

  public String toString() {
    return name + "," + quantity;
  }
}
```

```java
public class ColorItem extends Item {
  private String color;

  public ColorItem(String n,
      int q, String c) {
    Missing code 1
  }

  public String toString() {
    Missing code 2
  }

  public static void main(
      String[] args) {
    ColorItem ci = new ColorItem(
      "Apples", 3, "red");
    System.out.println(ci.toString());
  }
}
```

Missing code 1:

```java
super(n, q);
this.color = c;
```

Missing code 2:

```java
return super.toString() + "," + this.color;
```

Question 4. [10 points] For each of the following code fragments (a)–(d), state a big-O upper bound on the running time, with the problem size n being the value of the variable n. Briefly explain each bound.

(a)
```
int sum = 0;
for(int i = 0; i < n; i++) {        — n times
  for(int j = 0; j < n*n; j++) {    — n^2 times
    sum++;   O(1)
  }
}
```

$n \cdot n^2 \cdot O(1)$ is $O(n^3)$

(b)
```
int sum = 0;
for(int i = 0; i < n; i++) {
  for(int j = 0; j < i; j++){   ← avg. # of iterations is n/2
    sum++;         dependent
  }
}
```

$n \cdot n/2$ is $O(n^2)$

(c)
```
int sum = 0;
for(int i = 0; i < n; i++){  — n times
  sum++;  O(1)
}
for(int i = 0; i < n; i++){  — n times
  sum++;  O(1)
}
```

$2 \cdot (n \cdot O(1))$ is $O(n)$

(d)
```
int sum = 0;
for(int i = 0; i < n; i++){
  for(int j = 0; j < i*i; j++){   0, 1, 4, 9, 16, 25, ... (n-1)^2 iterations
    sum++;          dependent
  }
}
```

$\sum_{i=0}^{n-1} i^2 =$ [I wouldn't really expect you to know this ☺]

it's $O(n^3)$, Google "sum of first n squares" for an explanation

Question 5. [10 points]

(a) Consider the following program (which begins on the left and continues on the right):

```
public class Q5 {
  public static void mystery(
      ArrayList<Integer> a) {

    int n = a.size() / 2;
    while (n > 0) {              N/2 times
      Integer x = a.remove(0);   O(N)
      a.add(x);
      n--;
    }
  }

  public static void main(
      String[] args) {
    ArrayList<Integer> nums =
      new ArrayList<Integer>();

    for (int i = 0; i < 6; i++) {
      nums.add(i);
    }

    mystery(nums);

    for (Integer val : nums) {
      System.out.println(val);
    }
  }
}
```

What output is printed when the program is run?

3
4
5
0
1
2

(b) State a big-O upper bound on the running time of the `mystery` method in part (a). Let the problem size N be the number of elements in the `ArrayList` passed as the parameter. Explain briefly.

Call to a.remove(0) is $O(N)$, and the loop executes $N/2$ iterations.

$N/2 \cdot O(N)$ is $O(N^2)$

Question 6. [5 points] What output is printed by the following code (which begins on the left and continues on the right)?

```
public class Q6 {
  public static List<Integer>
    mystery(List<Integer> a) {

    TreeSet<Integer> s =
      new TreeSet<Integer>();
    s.addAll(a);

    List<Integer> result =
      new ArrayList<Integer>();
    for (Integer val : s) {
      result.add(val);
    }

    return result;
  }
```

```
  public static void main(
      String[] args) {
    List<Integer> x =
      new ArrayList<Integer>();

    x.add(9);
    x.add(5);
    x.add(8);
    x.add(1);
    x.add(3);
    x.add(5);

    List<Integer> y = mystery(x);
    for (Integer val : y) {
      System.out.println(val);
    }
  }
}
```

Output

1
3
5
8
9

Question 7. [5 points] Briefly explain why the following code will not compile, and how to fix the problem so that the code does compile (and correctly count the number of lines in the file whose filename is passed as the parameter):

HERE

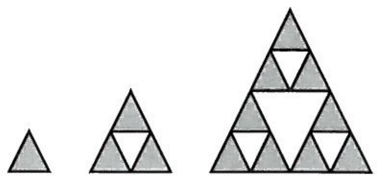
```
public static int countLines(String fileName) {
  FileReader fr = new FileReader(fileName);
  try {
    BufferedReader br = new BufferedReader(fr);
    int count = 0;
    while (br.readLine() != null) { count++; }
    return count;
  } finally {
    fr.close();
  }
}
```

throw checked exceptions (FileNot Exception and IOException)

To fix: at point labeled "HERE", add

throws IOException

Question 8. [5 points] Consider a fractal constructed in the following way. The first generation (0) of the fractal is an isosceles triangle whose base and height are each equal to 2. Each succeeding generation is constructed by connecting connecting three copies of the previous generation as shown below:



Generation: 0 1 2

Let $f(n)$ be the total area of all of the small (shaded) triangles in a generation n fractal. Our induction hypothesis, *IH*(n), is that for all $n \geq 0$, $f(n) = 2 \cdot 3^n$.

Basis step. Prove that *IH*(0) is true by showing that $f(0) = 2$. (The area of a triangle is $\frac{1}{2}bh$.)

$IH(0)$: $f(0) = 2 \cdot 3^0 = 2 \cdot 1 = 2$ ✓

By formula: area $= \frac{1}{2}bh = \frac{1}{2}(2 \cdot 2) = 2$ ✓

Expectation. State *IH*($n+1$): $f(n+1) =$ $2 \cdot 3^{n+1}$

Recurrence. Define $f(n+1)$ in terms of $f(n)$. In other words, how does the total area of the shaded triangles increase from one generation to the next? Explain briefly.

$f(n+1) = 3 \cdot f(n)$

because each subsequent generation includes 3 exact copies of the previous-generation fractal

Induction step. Show that if IH(n) is true, then *IH*($n+1$) must also be true. Expand the occurrence of $f(n)$ in your recurrence, and show that the resulting equation can be rewritten to exactly match the expectation.

$$
\begin{aligned}
f(n+1) &= 3 \cdot f(n) \\
&= 3^1 \cdot 2 \cdot 3^n \\
&= 2 \cdot (3^1 \cdot 3^n) \\
&= 2 \cdot 3^{n+1}
\end{aligned}
$$

✓ matches expectations