CIS 110 Fall 2015 — Introduction to Computer Programming
14 December 2015 — Final Exam
Answer Key

0.) The Easy One (1 point total)

Check cover sheet for name, recitation #, PennKey, and signature.

1.) Syntax (8 points total)

Answer each of the questions below.

1.1) (3 points) Circle the letter next to each of the following statements that is *true*:

(a) [✓ **The only methods a static method can call directly are other static methods.**]

(b) Every class must be public.

(c) Every instance variable must be private.

(d) [✓ **A class can have no constructors.**]

(e) [✓ **A class can have multiple constructors.**]

(f) Every class must have a main method.

1.2) (3 points) Circle the letter next to each pair of methods that can co-exist in the same class:

(a) [✓ `public void boo()` and `public int boo(int x)`]

(b) `public void boo()` and `public static void boo()`

(c) `public void boo(int x, int y)` and `public void boo(int c, int d)`

(d) [✓ `public void boo(int x, int y, String z)`] and [✓ `public void boo(String x, String y, String z)`]

(e) [✓ `public void boo()` and `public void Boo()`]

(f) `public void boo()` and `private int boo()`

1.3) (1 point) True or False: `mergesort` on an n-element linked list is faster than on an n-element array? False

1.4) (1 point) The Java expression `7 ^ 2` evaluates to: 5

2.) Sort of Sorted (9 points total)

For each of the arrays below, give the number of comparisons required to sort it using the specified algorithm.

2.1) (3 points) Using Selection Sort (Find the smallest element; swap it with the first element in the array; find the smallest remaining element; swap it with the select element; etc.)

(a) {8, 2, 1, 4, 3, 5} 15

(b) {1, 3, 8, 4, 2, 5} 15

(c) {1, 3, 5, 2, 8, 9} 15

2.2) (3 points) Using Insertion Sort (Swap each element with the one to its left as long as the one to its left is larger; start the process with the second element in the array and work up to the last element.)

(a) {8, 2, 1, 4, 3, 5} 10

(b) {1, 3, 8, 4, 2, 5} 10

(c) {1, 3, 5, 2, 8, 9} 7

2.3) (3 points) Using Merge Sort (Recursively merge sort each half of the array, then merge the two halves. *Split three elements [X Y Z] into a 2-element half [X Y] and a 1-element half [Z].)*

(a) {8, 2, 1, 4, 3, 5} 10

(b) {1, 3, 8, 4, 2, 5} 11

(c) {1, 3, 5, 2, 8, 9} 10

3.) Pugs and Other Toys (13 points total)
Arvind's pug, like all members of toy breeds, loves TOY. When Arvind returned from India, he wrote this TOY program just to celebrate. It reads n values from stdin (the first input value is n, just like on N-Body and TOY Encryption) and does something. But when Arvind asked what the program does, his pug just wagged his tail. The assembly comments are similar to X-Toy's, but not identical.

```
02: 0001   0000 0000 0000 0001
10: 8202   R[2] <- mem[02]
11: 6102   R[1] <- R[0] >> R[2]
12: 8FFF   R[F] = mem[FF]
13: 2FF2   R[F] <- R[F] - R[2]
14: DF18   if (R[F] >  0) goto 18
15: CF18   if (R[F] == 0) goto 18
16: _1FF   write R[1]
17: _000   halt
18: 8A__   read R[A]
19: CA1B   if (R[A] == 0) goto 1B
1A: 4121   R[1] <- R[2] ^ R[1]
1B: C013   goto 13
```

3.1) (3 points) Unfortunately the three instructions at memory addresses 16–18 got smeared by pug slobber, so you will need to complete them. Write each of the three completed instructions below.

(a) `16:` `91FF`

(b) `17:` `0000`

(c) `18:` `8AFF`

3.2) (7 points) For each of the lists of input values below, give the list of values the program will write to `StdOut`. The program will terminate properly, and will print *something*, in each of these cases.

(a) `{ 0 }:` 0

(b) `{ 1, 0 }:` 0

(c) `{ 1, 1 }:` 1

(d) `{ 4, 0, 1, 2, 3 }:` 1

(e) `{ 4, 0, 1, 0, 1 }:` 0

3.3) (3 points) If this program were a Java function, what would a reasonable name for it be?
`parityOfNonZeroValues`

4.) It's all good except the exceptions (7 points total)

Some of the following Java statements could cause one or more of the run-time errors listed below under certain circumstances. For each statement, write the letter(s) corresponding to the exception(s) it could trigger. If a statement could trigger more than one error, your answer should list multiple letters. Assume each statement compiles cleanly.

(a) `ArithmeticException`

(b) `ArrayIndexOutOfBoundsException`

(c) `StringIndexOutOfBoundsException`

(d) `NullPointerException`

(e) None

Statement	Exceptions
`int k = arr[0] / arr.length;`	abd -or- bd
`Color c = new Color(arr[0], arr[1], arr[2]);`	bd
`if (head.next != null) head = head.next;`	d
`if (s != null) return s.charAt(s.length());`	c
`double x = 5.0 / 0;`	e

5.) Cue Tickle (18 points total)

Elmo is a manicurist by day and pool shark by night. His hands are as deft as they are fantastic. No wonder that he's also into tickling, and most interactions with him can be modeled by the following Java program **on the next page**. In the space below, write the output that would be produced by running the following command at the interactions pane. **Draw a box around your answer so we know exactly what to grade!** *(Suggestion: Cross out the class, variable, and method names in the code, and replace them with meaningless names like `a`, `b`, and `c` before tracing the code.)*

```
java CueTickle Dont chip your nail on an eight ball
```

Your Answer

```
Dont: Boo!
Tickling chip
Tickling your
nail: Har!
eight: Hee!
Tickling ball
```

THE `CueTickle` CODE IS ON THE NEXT PAGE.

6.) Iterative Coding (10 points total)

For both this question and the following one, you should assume a well-formed linked list, where the last node's ***next*** *points to* ***null****. You do not need to perform any error checking or write any comments. Only write the requested method; do* ***not*** *write the surrounding class or add any instance variables.*
Consider this simple `ListNode` class.

```
public class ListNode {
    private int val;
    private ListNode next;
}
```

Add a `containsDups` method to the `ListNode` class that returns `true` if the list contains any duplicate values, and `false` otherwise. For example, `head.containsDups()` should return `true` if `head` is a `ListNode` instance that is the head of a linked list containing the values 3, 5, 3, 2, 0, but `false` if it is the head of a list containing 3, 0, 2, -1. This method must be ***iterative***; it may not make any method or function calls.

```
public boolean containsDups() {
    for (ListNode n = this; n.next != null; n = n.next) {
        int v = n.val;

        for (ListNode p = n.next; p != null; p = p.next)
            if (p.val == v) return true;
    }

    return false;
}
```

7.) Recursive Coding (10 points total) Add a `containsSum` method to the `ListNode` class from the previous question that takes an integer argument `total` and returns `true` if some combination of values in the list sums up to `total`. Otherwise it should return `false`. (The directions in italics at the top of the previous question apply to this one as well.)
For example `head.containsSum(3)` should return `true` if `head` is the head of a linked list containing the values 19, 2, -5, 1 (because $2 + 1 = 3$), but `false` if the list contains the values 1, 4, -5, 0. This function must be ***recursive***; it may not contain any loops. It can call itself recursively as many times as you wish, but it may not call any other methods. (You do not need any helper methods.)

```
public boolean containsSum(int total) {
    if (val  == total) return true;
    if (next == null)  return false;

    if (next.containsSumRecursive(total))       return true;
    if (next.containsSumRecursive(total - val)) return true;

    return false;
}
```