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

Name: _____

Recitation # (e.g., 201): _____

Pennkey (e.g., eeaton): _____

My signature below certifies that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this examination.

_____ _____
Signature Date

Instructions:

- **Do not open this exam until told by the proctor.** You will have exactly 120 minutes to finish it.
- **Make sure your phone is turned OFF (not to vibrate!) before the exam starts.**
- Food, gum, and drink are strictly forbidden.
- **You may not use your phone or open your bag for any reason**, including to retrieve or put away pens or pencils, **until you have left the exam room**.
- This exam is closed-book, closed-notes, and closed-computational devices.
- If you get stuck on a problem, it may be to your benefit to move on to another question and come back later.
- All code must be written out in proper Java format, including all curly braces and semicolons.
- Do not separate the pages. If a page becomes loose, reattach it with the provided staplers.
- Staple all scratch paper to your exam. Do not take any sheets of paper with you.
- If you require extra paper, please use the backs of the exam pages or the extra pages provided at the end of the exam. **Clearly indicate on the question page where the graders can find the remainder of your work (e.g., "back of page" or "on extra sheet")**.
- Use a pencil, or blue or black pen to complete the exam.
- If you have any questions, raise your hand and a proctor will come to answer them.
- When you turn in your exam, you may be required to show ID. **If you forgot to bring your ID, talk to an exam proctor immediately**.
- We wish you the best of luck. Have a great Fall break!

Scores: [For instructor use only]

Question 0		1 pts
Question 1		8 pts
Question 2		9 pts
Question 3		13 pts
Question 4		7 pts
Question 5		18 pts
Question 6		10 pts
Question 7		10 pts
Total:		76 pts

TOY Reference Card

INSTRUCTION FORMATS

```
            | . . . . | . . . . | . . . . | . . . . |
  Format 1: | opcode  |    d    |    s    |    t    |  (0-6, A-B)
  Format 2: | opcode  |    d    |        imm        |  (7-9, C-F)
```

ARITHMETIC and LOGICAL operations

```
    1: add              R[d] <- R[s] +  R[t]
    2: subtract         R[d] <- R[s] -  R[t]
    3: and              R[d] <- R[s] &  R[t]
    4: xor              R[d] <- R[s] ^  R[t]
    5: shift left       R[d] <- R[s] << R[t]
    6: shift right      R[d] <- R[s] >> R[t]
```

TRANSFER between registers and memory

```
    7: load immediate   R[d] <- imm
    8: load             R[d] <- mem[imm]
    9: store            mem[imm] <- R[d]
    A: load indirect    R[d] <- mem[R[t]]
    B: store indirect   mem[R[t]] <- R[d]
```

CONTROL

```
    0: halt             halt
    C: branch zero      if (R[d] == 0) pc <- imm
    D: branch positive  if (R[d] >  0) pc <- imm
    E: jump register    pc <- R[d]
    F: jump and link    R[d] <- pc; pc <- imm
```

Register 0 always reads 0.
Loads from mem[FF] come from stdin.
Stores to mem[FF] go to stdout.

0.) The Easy One (1 point total)

- Check to make certain that your exam has all 10 pages (excluding the cover sheet).
- Write your name, recitation number, and PennKey (username) on the front of the exam.
- Sign the certification that you comply with the Penn Academic Integrity Code.

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? _____

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

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} _____

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

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

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} _____

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

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

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} _____

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

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

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:` _____

(b) `17:` _____

(c) `18:` _____

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 }`: _____

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

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

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

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

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

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;`	-or-
`Color c = new Color(arr[0], arr[1], arr[2]);`	
`if (head.next != null) head = head.next;`	
`if (s != null) return s.charAt(s.length());`	
`double x = 5.0 / 0;`	

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

THE `CueTickle` CODE IS ON THE NEXT PAGE.

Cue Tickle (Code)

```
public class CueTickle {
    private class Nail {
        private String finger;
        private Nail   polish;
        Nail(String food, Nail sausage) {
            finger = food;
            polish = sausage;
        }
    }

    private Nail cue, eightBall;

    public void tickleMe(String where) {
        if (where.length() < 3) return;
        if (cue != null) {
            eightBall.polish = new Nail(where, null);
            eightBall = eightBall.polish;
            System.out.println("Tickling " + where);
        } else {
            eightBall = new Nail(where, null);
            cue = eightBall;
            System.out.println("Tickling " + where);
        }
    }

    public void laughAt(String atWhat) {
        if (eightBall == null) {
            System.out.println(atWhat + ": Boo!");
        } else if (cue != eightBall) {
            System.out.println(atWhat + ": Har!");
            cue = cue.polish;
        } else {
            System.out.println(atWhat + ": Hee!");
            cue = null;
            eightBall = null;
        }
    }

    public static void main(String[] args) {
        CueTickle elmo = new CueTickle();
        for (int i = 0; i < args.length; i++) {
            if (i % 3 == 0) elmo.laughAt(args[i]);
            else            elmo.tickleMe(args[i]);
        }
    }
}
```

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.

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.)

[**Scrap Paper: This Page Intentionally Blank**]