CIS 110 — Introduction to Computer Programming
20 December 2013 — 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 winter break!

Scores: [For instructor use only]

Question 0		1 pts
Question 1		8 pts
Question 2		12 pts
Question 3		11 pts
Question 4		23 pts
Question 5		12 pts
Question 6		18 pts
Total:		85 pts

TOY Reference Card

INSTRUCTION FORMATS

```
             | . . . . | . . . . | . . . . | . . . .|
  Format 1:  | opcode  |    d    |    s    |    t    |  (0-6, A-B)
  Format 2:  | opcode  |    d    |       addr        |  (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 address	R[d] <- addr
8: load	R[d] <- mem[addr]
9: store	mem[addr] <- 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 <- addr
D: branch positive	if (R[d] > 0) pc <- addr
E: jump register	pc <- R[d]
F: jump and link	R[d] <- pc; pc <- addr

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 12 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.) MISCELLANEOUS (8 points total)

1.1) (2 points) What is the common pattern of class definitions that we used in class?

(a) Methods and field variables are both public

(b) Methods are public, and field variables are private

(c) Methods are private, and field variables are public

(d) Methods and field variables are both private

1.2) (2 points) Suppose that `p` and `q` are both nodes in a linked list of strings. What happens when the expression `p.data == q.data` is evaluated?

(a) The expression is true if `p.data` and `q.data` refer to the same object instance

(b) The expression is true if `p.data` and `q.data` refer to objects with the same field values

(c) The expression is false

(d) Compiler error

(e) Run-time error

1.3) (2 points) Suppose that `p` and `q` are both nodes in a linked list of strings. What happens when the expression `p.data.equals(q.data)` is evaluated?

(a) The expression is true if `p.data` and `q.data` refer to the same object instance

(b) The expression is true if `p.data` and `q.data` refer to objects with the same field values

(c) The expression is false

(d) Compiler error

(e) Run-time error

1.4) (2 points) Which ordering is correct from fastest to slowest computational complexity?

(a) SinglyLinkedList.insert(0, x) $\leq$ SinglyLinkedList.insert(x) $\leq$ BinarySearchTree.insert(x)

(b) BinarySearchTree.contains(x) $\leq$ SinglyLinkedList.contains(x) $\leq$ contains(array, x)

(c) selection sort $\leq$ merge sort $\leq$ insertion sort

(d) remove(array, x) $\leq$ SinglyLinkedList.remove(x) $\leq$ BinarySearchTree.remove(x)

(e) None of the above are correct

(Assume that `contains(array, x)` determines whether an arbitrary (unsorted) `array` contains `x`, and that `remove(array, x)` removes `x` from the given arbitrary `array`.)

2.) OBJECT ORIENTED PROGRAMMING (12 points total)

```
public interface Animal {
    public String getName();
    public String speak();
}

public interface Talker {
    public String speak(String s);
}
```

```
public class Dog implements Animal {
    private String name = null;
    public Dog(String name) { ... }
    ...
}

public class CartoonCharacter
             implements Animal, Talker {
    public CartoonCharacter(String name) { ... }
    ...
}
```

2.1) (4 points) Are the following code fragments valid? (Yes or No)

(a) `CartoonCharacter c = new CartoonCharacter("Mickey Mouse"); Animal a = c; a.speak();`

(b) `Dog d = new Dog("Pluto"); CartoonCharacter c = (Animal) d; c.speak("Hello");`

2.2) (3 points) List the signatures of **all** methods that can be called on an instance of the `CartoonCharacter` class (excluding constructors).

2.3) (2 points) Provide the body for the Dog constructor, based on the class definition above.

```
public Dog(String name) {
  // your code here




}
```

2.4) (3 points) Briefly (**thirty words or less**) describe the purpose of an interface in Java (e.g., `Animal`).

3.) DEBUGGING (11 points total)

3.1) (6 points) Each 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 error(s) it could trigger. If a statement could trigger more than one error, your answer should list multiple letters.

(a) `java.lang.NullPointerException`

(b) `java.lang.ArrayIndexOutOfBoundsException`

(c) `java.lang.IllegalArgumentException`

(d) `java.util.InputMismatchException`

(e) `java.lang.RuntimeException`

______________________________ `linkedlist.get(10);`

______________________________ `vertices[i] = v;`

______________________________ `arr[i] = StdIn.readInt();`

______________________________ `if (node.next != null) tmp = node.next.next;`

______________________________ `if (node != null) tmp = node.next.next;`

______________________________ `if (vertices.length > i) vertices[i] = v;`

3.2) (1 point) When compiling your program, you receive the compiler error, "**missing return statement**." Which of the following could be the source of this error. Circle **<u>all</u>** that apply.

(a) A `private` method that should be `public`.

(b) A `void` method contains more than one `return` statement.

(c) A non-`void` method does not contain a `return` statement.

(d) There is a way to reach the end a non-`void` method without reaching a `return` statement.

(e) The compiler thinks there is a way to reach the end of a non-`void` method without reaching a `return` statement.

3.3) (1 point) When testing Guitar Hero, you receive a **java.lang.ArrayIndexOutOfBoundsException**. From the stack trace, you see that the error occurs at the following statement in `GuitarHero.java`:

```
strings[note].pluck();
```

Recall that `strings` is an array of `GuitarString` objects. Which of the following debugging strategies are most likely to help you pinpoint the error? Circle all that apply.

(a) Print out **note** immediately before this statement.

(b) Print out **strings** immediately before this statement.

(c) Print out **strings**[**note**] immediately before this statement.

(d) Print out **strings.length** immediately before this statement.

3.4) (3 points) After you find and correct the array error from Question 3.3, you receive a **java.lang.NullPointerException**, which you trace to the same statement. You discover it is occurring because **strings**[**0**] is **null**. In thirty words or less, explain the most likely cause of this.

4.) VIRTUAL BLING (23 points total)
This holiday season, you have decided to give virtual strands of beads to all your virtual Facebook friends. Each **Bead** has a color and a diameter. The **StrandOfBeads** class implements a virtual strand of beads as a doubly linked list of **Bead**s with a sentinel node at either end.

(Recall that with sentinel nodes, the first and last nodes in the list don't represent beads. Also, recall that in a doubly linked list, each node points to both the next node and the previous node.)

```
public interface Bead {
  public String getColor();
  public double getDiameter();
}

public class StrandOfBeads {
  private class Node {
    Bead bead;
    Node next;
    Node prev;

    Node(Bead b, Node n, Node p) {  bead = b;   next = n;   prev = p; }
  }

  private Node first;  // the sentinel node at the beginning of the strand
  private Node last;   // the sentinel node at the end of the strand

  // create an empty strand of beads
  public StrandOfBeads() {
    last = new Node(null, null, null);      // sentinel node for the tail
    first = new Node(null, last, null);     // sentinel node for the head
    last.prev = first;
  }

  // return true if the strand has no beads on it, false otherwise
  public boolean isEmpty() { /* IMPLEMENT THIS METHOD */ }

  // add a bead to the beginning/end of the strand
  // if the argument is null, do nothing
  public void addFirst(Bead b) { ... }
  public void addLast(Bead b)  { /* IMPLEMENT THIS METHOD */ }

  // remove and return the first/last bead from the strand
  // if the strand is empty return null
  public Bead removeFirst() { ... }
  public Bead removeLast()  { /* IMPLEMENT THIS METHOD */ }
} // end StrandOfBeads class
```

```
}
```

4.1) (3 points) Implement **isEmpty()** here:

```
public boolean isEmpty() {








}
```

4.2) (6 points) Implement **addLast()** here:

```
public void addLast(Bead b) {














}
```

4.3) (6 points) Implement **removeLast()** here:

```
public Bead removeLast() {














}
```

4.4) (8 points) For your many Penn friends, you want to make sure you give strands that don't include the Princeton colors `"black"` and `"orange"`. However, the manufacturer made a mistake and all the strands contain at least some black and orange beads.

Write a static function **pennify** that accepts a **StrandOfBeads** and returns a new **StrandOfBeads**. It should discard all beads whose color is `"black"` or `"orange"`, and move all other beads from the old strand to the new strand, while maintaining their order.

- Your method may modify the input **StrandOfBeads**.
- If the original strand is **null**, your method should return **null**.
- Assume that all the methods in **StrandOfBeads** are implemented correctly.
- Assume that your **pennify** function is not contained in the **StrandOfBeads** class, but do not write its surrounding class (only write the function).
- You are not required to write comments, but may use them to help clarify your code.

5.) TOY (12 points total) The following TOY program reads one value from standard input, performs a series of computations, and writes three values to standard output. The program and assembly language comments are correct as given.

```
01: FFFF   (1111 1111 1111 1111,     -1)
10: 7A01   R[A] <- 0001
11: 7B03   R[B] <- 0003
12: 8C01   R[C] <- mem[01]
13: 8DFF   read R[D]
14: 5EDB   R[E] <- R[D] << R[B]
15: 2EED   R[E] <- R[E] - R[D]
16: 9EFF   write R[E]
17: 4FEC   R[F] <- R[E] ^ R[C]
18: 1FFA   R[F] <- R[F] + R[A]
19: 9FFF   write R[F]
1A: 1FFE   R[F] <- R[F] + R[E]
1B: 9FFF   write R[F]
1C: 0000   halt
```

For the questions below, you may write all values the program prints as 4-digit hexadecimal numbers (e.g., 00A3), **or** as decimal (e.g., -623). However, you must not mix the two formats.

5.1) (3 points) What three values does this program write if it reads the value 0?

5.2) (3 points) What three values does this program write if it reads the value 1?

5.3) (3 points) What three values does this program write if it reads the value 2?

5.4) (3 points) In **twenty words or less**, what three values does this program compute? Do not tell us *how* it computes these values, only what they are intuitively. If you prefer, you may give your answer as formulas in terms of a value x that the program reads in.

6.) TREES (18 points total)

```
         14
       /    \
     7        11
    / \      /  \
   1   3    10   30
  / \      /    /
 0   2    7    42
```

6.1) (6 points) Circle all the true statements about the tree above:

(a) The tree is a binary tree.

(b) The tree is a binary search tree.

(c) The tree is complete.

(d) The tree is full.

(e) The tree has a height of 4.

(f) The node "30" is at depth 2.

6.2) (2 points) Draw a box around the root node and circle the leaf nodes in the diagram above.

6.3) (4 points) Label the following tree traversals as pre-order, in-order, post-order, or invalid. (Invalid signifies that one or more sequences are improper traversals of the tree.)

__________________________ `14 7 1 0 2  3 11 10  7 30 42`

__________________________ `0 1 2 3 7  7 10 11 14 30 42`

__________________________ `0 1 2 7 3 14  7 10 11 42 30`

__________________________ `0 2 1 3 7  7 10 42 30 11 14`

6.4) (2 points) Suppose that T is a binary tree with 13 nodes. What is the minimum possible height of T?

(a) 0

(b) 3

(c) 4

(d) 5

6.5) (4 points) Draw the binary search tree generated by the following operations:

insert: 4, 2, 1, 6, 5, 3 (in that order)

remove: 4

Scrap Paper

(This page is intentionally blank.)

Scrap Paper

(This page is intentionally blank.)