

CIS 110 — Introduction to Computer Programming
20 December 2013 — Final Exam

Answer Key

0.) THE EASY ONE: (1 point total) Check coversheet for name, recitation #, PennKey, and signature.

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) ≤ SinglyLinkedList.insert(x) ≤ BinarySearchTree.insert(x)`
- (b) **`BinarySearchTree.contains(x) ≤ SinglyLinkedList.contains(x) ≤ contains(array, x)`**
- (c) `selection sort ≤ merge sort ≤ insertion sort`
- (d) `remove(array, x) ≤ SinglyLinkedList.remove(x) ≤ BinarySearchTree.remove(x)`
- (e) None of the above are correct

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();`
[Yes](#)
- (b) `Dog d = new Dog("Pluto"); CartoonCharacter c = (Animal) d; c.speak("Hello");`
[No](#)

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

[getName\(\), speak\(\), speak\(String s\)](#)

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

```

public Dog(String name) {
    // your code here
    this.name = name;
}

```

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

[Answer must mention something about formalizing an API, or requiring that classes implement methods in the interface.](#)

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`

A, C	<code>linkedlist.get(10);</code>
A, B	<code>vertices[i] = v;</code>
A, D	<code>arr[i] = StdIn.readInt();</code>
A	<code>if (node.next != null) tmp = node.next.next;</code>
A	<code>if (node != null) tmp = node.next.next;</code>
A, B	<code>if (vertices.length > i) vertices[i] = v;</code>

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 all 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. **The `strings` array was created but the individual `GuitarString` objects were not created.**

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 **Beads** 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() {
    return first.next == last; // an empty list has only the two sentinel nodes
}
```

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

```
public void addLast(Bead b) {
    if (b == null) return;
    Node n = new Node(b, last, last.prev); // create new node to insert at end
    last.prev.next = n;                  // stick it just before the end sentinel
    last.prev = n;                       // update the end sentinel's prev pointer
}
```

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

```
public Bead removeLast() {
    if (isEmpty()) return null;

    Bead b = last.prev.bead; // get the last bead

    last.prev.prev.next = last; // drop the last bead's node from the list
    last.prev = last.prev.prev;

    return b;
}
```

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.

```
public static StrandOfBeads pennify(StrandOfBeads strand) {
    if (strand == null) return null;

    StrandOfBeads newStrand = new StrandOfBeads();

    while (!strand.isEmpty()) {
        Bead b = strand.removeFirst();
        String color = b.getColor();
        if (!color.equals("orange") && !color.equals("black"))
            newStrand.addLast(b);
    }

    return newStrand;
}
```

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?
 0, 0, and 0 *or* 0000, 0000, and 0000

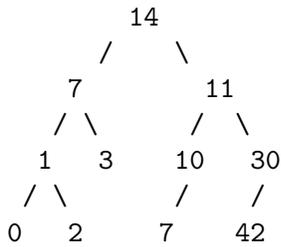
5.2) (3 points) What three values does this program write if it reads the value 1?
 7, and -7, and 0 *or* 0007, FFF9, and 000

5.3) (3 points) What three values does this program write if it reads the value 2?
 14, -14, and 0 *or* 000E, FFF2, and 0000

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.

$7x$, $-7x$, and 0

6.) TREES (18 points total)



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.)

preorder	14 7 1 0 2 3 11 10 7 30 42
invalid	0 1 2 3 7 7 10 11 14 30 42
inorder	0 1 2 7 3 14 7 10 11 42 30
postorder	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)

```

      4
     / \
    2   6
   /\  /
  1 3 5
    
```

remove: 4

```

      3          5
     / \      / \
    2   6    2   6
   /   /    /\
  1   5    1  3
    
```