CS61B, Fall 2011 **Final Examination (corrected)** **P. N. Hilfinger**

READ THIS PAGE FIRST. Your exam should contain 16 problems on 16 pages. Officially, it is worth 50 points.

This is an open-book test. You have three hours to complete it. You may consult any books, notes, or other inanimate objects available to you. You may use any program text supplied in lectures, problem sets, or solutions. Please write your answers in the spaces provided in the test. Make sure to put your name, login, and lab section in the space provided below. Put your login and initials *clearly* on each page of this test and on any additional sheets of paper you use for your answers.

Be warned: my tests are known to cause panic. Fortunately, this reputation is entirely unjustified. Just read all the questions carefully to begin with, and first try to answer those parts about which you feel most confident. Do not be alarmed if some of the answers are obvious. Should you feel an attack of anxiety coming on, feel free to jump up and run around the building once or twice.

Your name: ____________________ Login: __________

1. ______/4	9. ______/3	Login of person to your left: ______________
2. ______/3	10. ______/4	Login of person to your right: ______________
3. ______/5	11. ______/3	Discussion section number or time: ______________
4. ______/2	12. ______/4	Discussion TA: ______________
5. ______/2	13. ______/2	Lab section number or time: ______________
6. ______/	14. ______/2	Lab TA: ______________
7. ______/6	15. ______/2	
8. ______/4	16. ______/4	
	TOT ______/50	

Reference Material

```
/* From java.lang.System: arraycopy */
/** Assumes that SRC and DEST are arrays of the same type. Sets LENGTH
 *  elements of DEST, beginning at DEST[DESTPOS] to the initial
 *  contents of SRC, beginning at SRC[SRCPOS].  Raises
 *  IndexOutOfBoundsException if any of arguments is negative or if
 *  the request would copy non-existent elements. */
public static void arraycopy(Object src, int srcPos, Object dest, int destPos,
                             int length);
```

1. [4 points]

a. The following code is incorrect. How would you correct it?

```
class MyClass {
    /** Append the contents of ARRAY2 to the end of ARRAY1. */
    public static void appendArrays(int[] array1, int[] array2) {
        if (array1 == null)
            throw new IllegalArgumentException();
        if (array2 == null)
            throw new IllegalArgumentException();
        int[] array = new int[array1.length + array2.length];
        System.arraycopy(array1, 0, array, 0, array1.length);
        System.arraycopy(array2, 0, array, array1.length, array2.length);
        array1 = array;
    }
}
```

b. This code compiles. What is the output of executing it?

```
class C {
    public static void main(String[] args) {
        ArrayList L = new ArrayList<String>();
        L.add(new Integer(0));
        System.out.println(L.get(0));
    }
}
```

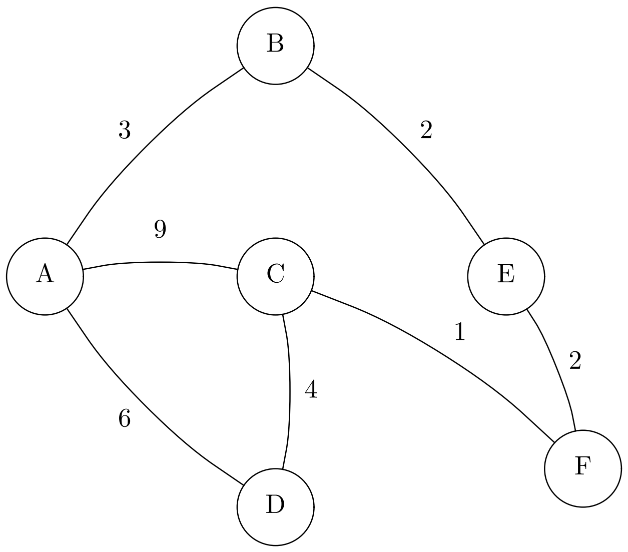
2. [3 points] We are working on a digital signal processing application and we need a very compact representation of positive integers over a wide range of values, even at the expense of precision, so we are going to design our own representation. We'll use one byte for each integer. The leading 4 bits will be a coefficient and the trailing 4 bits will be an exponent of 16. For example, `0x20` (`00100000` in binary) will represent $2 \times 16^0 = 2$ in ordinary decimal format, and `0x97` (binary `10010111`) will represent $9 \times 16^7 = 2415919104$. Implement multiplication for this number system. Your implementation language is a dialect of Java in which the largest integer type is `short` (16 bits). For example, multiplying the two sample numbers above would yield $2 \times 9 \times 16^{7+0} = 18 \times 16^7 = 9 \times 16^8$, represented `0x98`. You will have to do some rounding, due to the limited precision of the format. For simplicity, round down (throw away remainders). For example, $(11 \times 16^2) \times (10 \times 16^3) = 121 \times 16^5 = (7 + 9/16) \times 16^6 \approx 7 \times 16^6$, represented `0x76`. Hint: The program is short and contains no loops.

```
byte mult4(byte x, byte y) { // FILL IN









}
```

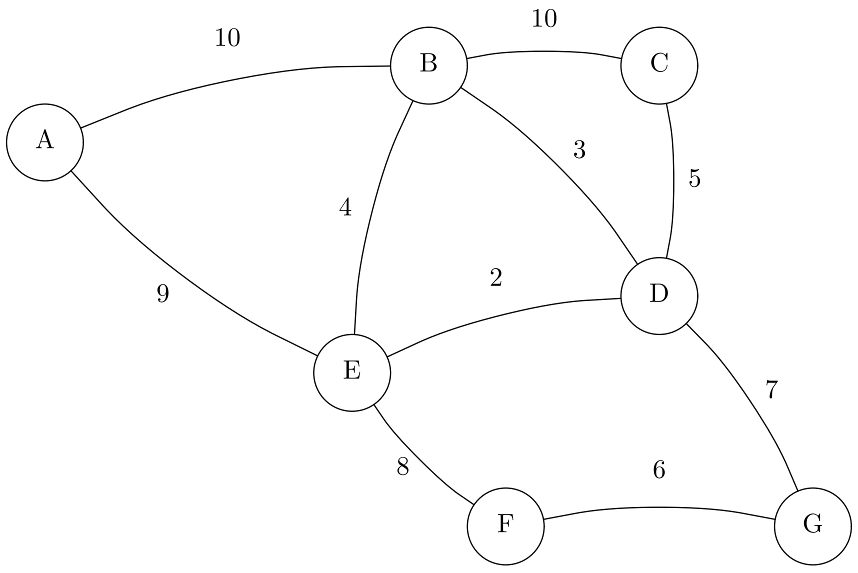
3. [5 points] Fill in the definition of `isBalanced` below to fulfill the comment. Feel free to write as many auxiliary functions as needed.

```
class BinaryTree {
    /** Returns my left child (null for an empty child). */
    BinaryTree getLeft() { ... }
    /** Returns my right child (null for an empty child). */
    BinaryTree getRight() { ... }
    ...
    /** Returns true iff T is perfectly balanced (all nodes have left
     *  and right children of identical shape). */
    public static boolean isBalanced(BinaryTree t) {














    }
}
```

4. [2 points] Suppose we run Dijkstra's algorithm on the following graph, starting from vertex A. In what order does the algorithm remove vertices from its queue, thus determining their final distances?



5. [2 points] Suppose we run Kruskal's algorithm on the following graph. What edges does the algorithm add to the tree and in what order? (Identify edges by their end points. For example, "AB").



6. [1 point] What is approximated by the following sequence?

$$\begin{aligned} a_0 &= 0.4 \\ a_k &= 0.4 + 0.3 \cdot 2^{k-1},\ k > 0 \end{aligned}$$

7. [6 points] Answer "true" or "false" for each of the following statements and give a *short* explanation. for each answer. *IMPORTANT:* you *must* give an explanation for each to get credit!

a. Depth-first search is a linear-time algorithm.

b. Suppose G is a graph with positive numbers as edge weights. If we add the same positive number to the weight of every edge, we will not change the shortest-path tree (merely the lengths of the shortest paths).

c. Every weighted graph has a unique minimum spanning tree.

d. Given N integers $a_1, \ldots, a_n$, it is possible to find the 100^{th} smallest number in $O(n)$ time.

e. A preorder tree walk on a max heap that prints the values it encounters will output the values stored in the heap in reverse sorted order.

f. Open address hash tables are more compact and therefore preferable to chaining as long as the load factor is less than 1.

8. [4 points] Provide short answers to the following questions.

a. Why does Dijkstra's algorithm not work for graphs with negative cost edges?

b. Given an array of N elements that have keys that are either -1, 0, or 1, describe an algorithm to sort the array in $O(N)$ time.

9. [3 points] Consider the following pseudo-code algorithm for sorting an array of numbers (`merge` is the same function used in mergesort):

```
trisort(A[0..N-1]):
    if (N <= 1) {
        return A;
    }
    B = trisort(A[1..N/3])
    c = trisort(A[N/3+1..2N/3])
    d = trisort(A[2N/3+1..N])
    bc = merge(B,C)
    bcd = merge(BC, D)
    return bcd
```

What is the running time of the algorithm? Explain.

10. [4 points] Match the following sorting algorithms to the appropriate data set by filling in four of the blanks. You may only use each algorithm once, so be sure that you are picking the best possible algorithm for that set. Not all algorithms will be used. Justify your responses in one or two sentences; answers without justification (or with incorrect justifications) will not be given any credit.

- Insertion Sort ______________
- Quick Sort ______________
- Radix Sort ______________
- Counting Sort ______________
- Heap Sort ______________

The data sets:

a. Sorting an array of 100,000,000 integers between 0 and 32,000.

b. Independently sorting 500,000 arrays, each with 5 elements.

c. Sorting a mission-critical set of $> 7,000,000$ numbers in worst case $\Theta(n \lg n)$ time.

d. Sorting 100,000,000 data values, which you can compare, but cannot examine directly

11. [3 points] How long does it take to compute each of the following? Give a *brief* explanation of each.

a. The maximum depth of an arbitrary binary search tree.

b. The maximum element of an arbitrary binary search tree.

c. The k^{th} largest element of an arbitrary binary search tree.

d. The maximum depth of a (2,4) tree.

e. The maximum element of (2,4) tree.

f. The k^{th} largest element of a (2,4) tree.

12. [4 points] The *segmentation problem* is to segment text that is written without spaces into individual words. For example, if you are given the text “thisexamishard”, the best segmentation is “this exam is hard”, not “thi sex am is hard’ or “thisex am ishard.” Assume you are given a (black-box) quality function q that takes a string of letters w and returns in constant time a value $q(w)$ that gives a measure of how likely w is to be an English word. The quality of w can be positive or negative so that $q($`"this"`$)$ is positive, and $q($`"thisex"`$)$ is negative. Given a string s, a *segmentation* of s is a partition of the letters into contiguous blocks of letters. The *total quality* of a segmentation is the sum of the quality of each of the blocks. So the quality of our segmentation above is $q($`"this"`$) + q($`"exam"`$) + q($`"is"`$) + q($”`hard`”$)$. Give an efficient algorithm that takes as input a string s and returns the maximum total quality achievable by a segmentation of s. Your algorithm must be polynomial in the length of s. We’re interested in the algorithm; pseudo-code is fine. Give an asymptotic time bound for your algorithm.

13. [2 points] Here is a function for computing Fibonacci numbers. It is the standard recursive routine, augmented to cache previously computed values, but it only caches values for odd n:

```
HashMap<Integer, Integer> cache = new HashMap<Integer, Integer>();
int fib(int n) {
    if (n <= 1) return n;
    if (n % 2 == 0) {
        return fib(n-1) + fib(n-2);
    } else if (!cache.contains(n)) {
        cache.put(n, fib(n-1) + fib(n-2));
    }
    return cache.get(n);
}
```

a. Draw a complete recursion tree `fib(7)`. (In a recursion tree, the nodes are labeled with calls to `fib` and the children of a node `fib(`x`)` are the recursive calls made during that call.)

b. What is the big-O running time of this routine? Explain your answer

14. [2 points] Bob is trying to come up with a regular expression that matches only valid amounts of money (in dollars and cents). For our purposes, a valid dollar amount must start with a dollar sign and be a positive number. An amount without cents must not include a decimal point,and those that do include cents must have exactly 2 decimal places. In addition, dollar amounts less than 1 must have exactly one zero before the decimal point. The regular expression he has come up with is `\$[0-9]*\.[0-9]*` (as a Java string literal: `"\\$[0-9]*\\.[0-9]*"`).

a. His pattern does not really work. Find two examples of ill-formed dollar amounts that would be accepted by this regular expression.

b. Rewrite the regular expression to solve these problems.

15. [2 points] In an attempt to implement the trip client of Project 3, Bob writes the following classes to represent roads and their names:

```
package trip;
import graph.Vertex;

public class Road<String, RoadName> extends graph.Edge<String, RoadName>{
    protected Road(Vertex<String> v0, Vertex<String> v1,
                   RoadName label, double weight, String direction) {
        super(v0, v1, label);
        ...
    }
    public String getDirection() {
        return this.getLabel().getDirection();
    }
}
------------------------------------
package trip;

public class RoadName {
    private String name;
    private String direction;

    public RoadName(String name, String direction) {
        this.name = name;
        this.direction = direction;
    }
    public String getDirection() {
        return direction;
    }
}
```

Aside from the lack of comments, the Road class does not compile. State why, and rewrite any lines that need to be fixed so that it does. [The change is not very big, so if you find yourself rewriting a lot of code, youre doing it wrong!]

16. [4 points] Fill in the following class to meet its comment. A method call `T.rearrange(x)` must run in time $O(C_s(T, x))$, regardless of N, the size of T. Here, $C_s(T, x)$ is the time required to search T for x. (This means that you will not be able to solve the problem by reconstructing the tree with N insertions, for example).

```
/** A binary search tree whose node labels are integers. */
class IntBST {
    /** Returns my label. */
    int getLabel() { ... }
    /** Returns my left child (null for an empty child). */
    IntBST getLeft() { ... }
    /** Returns my right child (null for an empty child). */
    IntBST getRight() { ... }
    /** Returns the result of rotating me to the right, returning the new
     *  top node. Runs in constant time. */
    IntBST rotateRight() { ... }
    /** Returns the result of rotating me to the left, returning the new
     *  top node. Runs in constant time. */
    IntBST rotateLeft() { ... }

    /** Modifies me to be a BST with the same set of labels, but with
     *  label X at the top of the tree. Assumes that X is in the
     *  tree. */
    public IntBST rearrange(int x) {















    }
}
```

Extra paper: