Your exam should contain 7 problems on 13 pages. Officially, it is worth 50 points.

This is an open-book test. You have three hours in which to complete it. You may consult any books, notes, calculators, or other inanimate objects (other than computers) 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.

Read all the questions carefully to begin with, and first try to answer those parts about which you feel most confident.

Your name: ____________________________________________ Login: __________

Login of person to your left: __________ Login of person to your right: __________

Discussion section number or time: __________ TA: __________

1. ________/12 5. ________/5
2. ________/10 6. ________/6
3. ________/10 7. ________/7
4. ________ TOT ________/50
1. [12 points] Answer each of the following briefly. Where a question asks for a yes/no answer, give a brief reason for the answer (or counter-example, if appropriate).

a. If \( f(x) \in \Theta(x^3) \) and \( g(x) \in O(x^2) \), and if there is some \( x_0 \) such that \( f(x_0) > g(x_0) \), then is \( f(x) > g(x) \) for all \( x > x_0 \)? Assume \( f \) and \( g \) are everywhere positive.

b. If \( g(x) = x^2 \cos x \), is \( g(x) \in O(x^2) \)? Is \( g(x) \in \Omega(x) \)?

c. A sorted list of values is maintained as a Java Vector whose initial capacity is \( N_0 \). That is, the representation consists of an array (initially of length \( N_0 \)) and a current size (always less than or equal to the current length of the array), and the array is expanded by factors of two as needed. What are the tightest asymptotic bounds you can give for the best and worst-case times for adding \( N = K \cdot N_0 \) values to this list (inserted in the right place to keep the list ordered), assuming the list is initially empty? (The “tightest” bound means “a \( \Theta(\cdot) \) bound if possible, and otherwise the smallest \( O(\cdot) \) bound and largest \( \Omega(\cdot) \) bound possible.”) We want bounds for the worst-case time and bounds for the best-case time. Include brief descriptions of the best and worst cases.
d. Suppose that I take a sorted linked list of $N$ integers, and break it into a list of $M$ equal-sized sorted lists of integers (that is, put the first $N/M$ integers into the first list, the next $N/M$ into the second, etc.). What is the worst-case time for finding whether an integer $x$ is in anywhere in this list of lists? If for some fixed $N$, you can choose $M$, what $M$ should you choose for maximum lookup speed?

e. You are in the process of testing your project and suddenly realize that you need to find out when and where a certain CS final is being held, but you have no friends in the course, you have no written information on the subject, and the professor and TAs refuse to tell you on the grounds that you should be able to find out for yourself. What do you do? Please be precise.

f. Consider a directed graph that has $V$ nodes and an in-degree of 2 (that is, no node has more than two incoming edges). What is the tightest bound you can give for the worst-case time to topologically sort this graph?
2. [10 points] The following fragment of Java code (which continues on the next page) contains several errors in some of the functions marked ‘// ?’.

a. [6 points] Find and (neatly) correct as many as possible, making as little change as possible. Do not correct things unless they really don’t work. WARNING: one of the marked functions is error-free! There may be more than one correction possible; be sure to choose one that makes the resulting program conform to the comments. Not all methods are shown; you don’t need to add any to correct the program.

```java
/** A Mapper represents a function from values to DLists.
 * If M is a Mapper, it is applied to a value, x, with M.map(x). */
public interface Mapper {
    DList map(Object x);
}

/** A DList represents a list of Objects. */
public class DList {
    /** The rep field of a DList points to a sentinel node in a circularly
     * linked list. Because the list is circular, the node after the
     * sentinel contains the first item in the list, and the node
     * before the sentinel contains the last item. For an empty DList,
     * the sentinel is its own next and previous element. Example of
     * list containing items A and B:
     * rep:  
     *       A --- B
     */

    /** An empty DList. */
    public DList() {
        rep = new DLink(null); rep.next = rep.prev = rep;
    }
    /** True iff THIS is an empty list. */
    public boolean empty() { return; } // ?
    /** The first item in THIS list. */
    public Object first() { return rep.next.value; }
    /** The last item in THIS list. */
    public Object last() { return rep.prev.value; }
    /** Add X to the beginning of THIS list. */
    public addFront(Object x) { // ?
        DLink item = new DLink(x);
        rep.next.spliceAfter(item, item);
    }
    /** Add X to the end of THIS list. */
    public addBack(Object x) { // ?
        DLink item = new DLink(x);
        rep.prev.spliceAfter(item, item);
    }
}
```

Continued
/** Remove the first item in THIS list. Requires !empty (). */
public void removeFront () { rep.next.unlink (); }

/** Remove the last item in THIS list. Requires !empty (). */
public void removeBack () { rep.prev.unlink (); }

/** Remove each value, x, that is currently in THIS list and insert
* in its place all the items in the list returned by M.map (x),
* concatenating the values returned by M.map end-to-end. Destructive:
* the DLists returned by M.map and original contents of THIS are modified.
* Is guaranteed to send items to M.map in their original order.
* For example, if M.map (x) returns the list [x, x] for any x,
* and L contains ["Hello", "world"], then L.mapcat (M) causes L
* to become [ "Hello", "Hello", "world", "world" ]. If M.map (x)
* always returns the empty list, then L.mapcat (M) makes L empty. */
public void mapcat (Mapper m) { // ?
  DLink d;
  d = rep;
  while (d != rep) {
    DLink n = d.next, p = d.prev;
    DList r = m.map (d.value);
    d.unlink ();
    p.spliceAfter (r.rep.next, r.rep.prev);
    d = n;
  }
}

private DLink rep;
}

class DLink {
/** The next and previous DLinks in the list. */
DLink next, prev;
/** The item stored in this link. */
Object value;
/** A DLink whose value is x. */
DLink (Object x) { value = x; next = prev = null; }

/** Remove THIS from the DList it is part of.
* Does not change THIS. */
void unlink () { prev = next.prev; next = prev.next; } // ?

/** Insert the list of items that starts at START and ends at
* END immediately after THIS. Requires START and END to be
* non-null. */
void spliceAfter (DLink start, DLink end) { // ?
  end.next = next; start.prev = this;
  next.prev = end; next = start;
}
}
b. [4 points] Suppose that DL is a DList of Strings. Somewhere in the middle of this list is the String "/". Given a String pivot, I want to remove all Strings that come before the "/" and are less than pivot (in dictionary order) and move them to immediately after the "/", leaving everything else after the "/" unchanged. Thus, if pivot is “quark” and DL contains [dog, bat, robot, ant, zebra, gnu, /, turkey, finch]. I want to end up with DL containing [dog, bat, ant, gnu, /, robot, zebra, turkey, finch]. I want the following expression to do this:

DL.mapcat (__________________________________________________);

How should you fill in the blank and what additional declarations should you add to get this effect?
3. [10 points] Consider a system that models a network of resistors and batteries. Each resistor has a resistance (a number) and each battery has a voltage (we oversimplify). Each resistor and battery has two terminals (which we'll call terminal 0 and terminal 1) to which wires may be connected. In Java, we might have class declarations like the following.

```java
/** A network resistors and batteries connected by wires. */
class Circuit {
    /** An empty network. */
    public Circuit () { ... }

    /** Add a new battery with voltage V, giving it the label NAME. */
    public void battery (String name, double V) { ... }

    /** Add a new resistor with resistance R, giving it the label NAME. */
    public void resistor (String name, double R) { ... }

    /** Connect a wire between terminal #K0 (0 or 1) of the component
     * named C0 and terminal #K1 of the component named C1.
     * Both C0 and C1 must have been defined with method battery
     * or resistor. */
    public void connect (String c0, int k0, String c1, int k1) {...}

    /** True iff there is a short in the system. This is defined as
     * a sequence of terminals t1, t2, t3, ..., tn, such that t1 and tn
     * are the two opposite terminals of the same battery, and for
     * each pair of terminals ti and ti+1, either ti and ti+1 are connected by a wire
     * or they are opposite terminals of the same battery. */
    public boolean isShort () { ... }

    // ADDITIONAL DECLARATIONS OF MEMBERS AND CLASSES (if any)
}
```

For example, the statements on the left below give the Circuit diagrammed on the right.

```java
Circuit C = new Circuit();
C.battery("B1", 9.0);
C.battery("B2", 3.0);
C.resistor("R1", 1e5);
C.resistor("R2", 1e6);
C.connect("B1",0,"R1",0);
C.connect("B1",0,"R1",1);
C.connect("R1",0,"R2",0);
C.connect("R1",1,"R2",1);
C.connect("R1",0,"B2",0);
C.connect("R1",1,"B2",1);
// NOTE: there are two "shorts":
// terminal 0 of B1 to 1 of R1 to 1 of B1;
// terminal 0 of B1 to 0 of R1 to 0 of R2
// to 0 of B2 to 1 of B2 to 1 of R2 to
// 1 of R1 to 1 of B1
```
a. [3 points] Show how to fill in the “ADDITIONAL DECLARATIONS” of Circuit as needed to provide the necessary representation for these classes. You may assume that only the member functions indicated need to be implemented (but you may add additional member functions if you find it necessary to do so).

Additional declarations for Circuit, if any:
b. [4 points] Provide bodies for the member functions `battery` and `connect`. Do not bother with error checking.

```java
public void battery (String name, double V) {
}
```

```java
public void connect (String c0, int k0, String c1, int k1) {
}
```

c. [3 points] Provide a body for member function `isShort`. You may use pseudo-code to shorten things as long as you are reasonably detailed. Your solution should be as fast as possible. Do not bother with error checking.

```java
public boolean isShort () {
}
```
4. [1 point] Name a poet much of whose work can be sung to the tune of *The Yellow Rose of Texas*.

5. [5 points] Dijkstra’s algorithm is not the only way to find the lengths of shortest paths in a graph whose vertices are numbered 1 to $n$. Let $w_{ij}$ be the cost (or weight) of the edge between $i$ and $j$ ($=\infty$ if there is no such edge). Suppose we define $\text{dist}(i, j, m)$ to be the length of the shortest path between vertex number $i$ and vertex number $j$ in the graph that contains no more than $m$ edges (steps). This leads to a simple recursive definition for $\text{dist}$:

$$
\text{dist}(i, j, 0) = \begin{cases} 
0, & \text{if } i = j \\
\infty, & \text{otherwise.} 
\end{cases}
$$

$$
\text{dist}(i, j, m) = \min_{1 \leq k \leq n} \{ \text{dist}(i, k, m - 1) + w_{kj} \}, \text{ for } m > 0
$$

That is, we look at the length of each path starting at $i$ that is one shorter than $m$ plus the cost of the edge from the end of that path to vertex $j$. Since this has to include every path between $i$ and $j$, the minimum of those sums must be the answer we want for the shortest path between $i$ and $j$. If you implemented this algorithm directly, how long would it take to compute $\text{dist}$? Describe a simple step you could take to speed it up, and tell how fast the result would be.
6. [6 points] Suppose that an array \( PQ \) contains a priority queue, represented by a heap (that is, the array represents a tree satisfying the heap property, with the children of the node stored in element \( i \) of the array being at positions \( 2i + 1 \) and \( 2i + 2 \)). Suppose we allow the user of the priority queue to modify the priority of any node in the heap up or down. What must we do to re-establish the heap property after such a modification? Use either Java or precise pseudo-code to give your answer (being sure to describe, at least in precise English, what each primitive operation you use means).
7. [7 points] The Box class below is to be used in a text-formatting program to represent a rectangular area of the page containing text. The call `Box.glyph(c, w, h)` where `c` is a character, and `w` and `h` are integers produces a simple Box containing just a single character and having width `w` and height `h`. The call `Box.hbox()` produces an initially empty “horizontal box” – a sequence of smaller Boxes that are glued together from left to right. Finally, `Box.vbox()` produces an initially empty “vertical box” – sequence of smaller Boxes that are glued together from top to bottom. You can add more Boxes to the right of a horizontal box or top of a vertical box using the `add` method. Every box has a height and width. The height and width of a glyph are specified in the call that creates it. The width of a horizontal box is the sum of the widths of the boxes added to it, and its height is the maximum of the heights of those boxes. The height of a vertical box is the sum of the heights of the boxes added to it and the width is the maximum of the widths of the boxes inside.

Using the definitions above, fill in the methods and classes below to obey the comments. Feel free to use any library methods and classes you want (you will find `Math.max` particularly useful). You may not add any conditional or looping statement—no `if`, `while`, `for`, `switch`, or `try` statements or conditional (?:) expressions anywhere in your solution. Do not use `instanceof`. **WARNING:** You will have missed the entire point of this problem and will get no credit if you violate these restrictions!!!

```java
public abstract class Box {
    /** A glyph box containing C, with width W and height H. */
    public static Box glyph(char c, int w, int h) {
        return ;
    }

    /** An empty horizontal box, initially of width and height 0. */
    public static Box hbox() {
        return ;
    }

    /** An empty vertical box, initially of width and height 0. */
    public static Box vbox() {
        return ;
    }

    /** The width of THIS. */
    public abstract int width();

    /** The height of THIS. */
    public abstract int height();

    /** Add B to the end (right or bottom) of THIS box. This operation */
    /** is undefined for glyphs. */
    public abstract void add(Box b);
}

// FILL IN ADDITIONAL CLASSES HERE (IF NEEDED) ON THE NEXT PAGE
```
Final Login: __________  Initials: ________

// FILL IN ADDITIONAL CLASSES HERE (IF NEEDED) HERE