NAME: ________________________________ NETID: ________________________

CS2110 Fall 2010 Final Exam
December 16, 2010, 2pm-3:30pm

Write your name and Cornell netid. There are 5 questions on 10 numbered pages and one extra credit problem on page 11 for a possible 6 points of extra credit (not to mention wealth beyond your dreams of avarice, as you'll see…).

- *Check now that you have all the pages.*
- *Write your answers in the boxes provided.*
- *Use the back of the pages for workspace.*
- *Ambiguous answers will be considered incorrect.*
- *Concise but correct answers will be considered correct. Answers that ramble on and make numerous false statements will lose points.*
- *The exam is closed book and closed notes.*
- *Some of the code is abbreviated to save space, or obfuscated to avoid making the problem too easy. This does not make it good coding style.*
- *If you are colorblind and don't want us to grade in a certain color ink, please make note of it at the top of every page.*
- ***Do not begin until instructed.***

You have 90 minutes. Good luck!

	1	2	3	4	5	Extra	Total
Score	/20	/16	/22	/20	/22	/6	
Grader							

1.(20 points) Sort Algorithms

(a) *(6 points)* There are three fundamental steps to the Quicksort algorithm. What are they?

1.

2.

3.

(b) *(5 points)* Quicksort has an average case runtime of $O(n \log n)$, but a worst case complexity of $O(n^2)$. Is this an issue in practice? Explain.

(c) *(4 points)* Suppose you want to sort an array of (x,y) pairs in lexicographic order, i.e., sorting by the x dimension and breaking ties with the y dimension.
Instead of writing a method to compare pairs directly, you make two calls to an existing function `DIMSORT(pair_array, dimension)`, which sorts a pair array in-place by an indexed dimension:

```
DIMSORT(pairs, 1);   // sort by Y dimension
DIMSORT(pairs, 0);   // sort by X dimension
```

This can work, but what assumption does it make about the implementation of `DIMSORT`?

(d) *(5 points)* Here is pseudo-code for **SLOWSORT**, notable for being a truly awful sorting algorithm, even in the best case:

```
function SLOWSORT(int[] A, int i, int j):
    if i >= j: return
    m = (i + j) / 2
    SLOWSORT(A, i, m)
    SLOWSORT(A, m + 1, j)
    if A[j] < A[m]:
        swap(A[j], A[m])
    SLOWSORT(A, i, j - 1)
```

d-i) (3 points) Let R(n) be the runtime of **SLOWSORT** on an array A of size n, measured as the number of times **SLOWSORT** is called. Write R(n) as a recursive expression, but don't try to simplify. *To simplify, assume n>2. Use the notation* $\lceil exp \rceil$ *to denote exp "rounded up" and* $\lfloor exp \rfloor$ *to denote exp "rounded down", where exp is an expression.*

R(n) =

d-ii) (2 points) Normally we measure complexity for sorting in terms of the number of comparison/swap operations performed. Is this the same as R(N) from above? Explain.

2. (16 points) True or false?

Parts a,b,c, and d all refer to the same code:

```
try { stmt } catch(E1) { stuff1 } catch (E2) { stuff2 } finally {F}
```

a	T	F	An exception of type E1 occurs while evaluating `stmt`. This causes **`stuff1; F`** to be executed.
b	T	F	An exception of type E2 occurs while evaluating `stmt`. This causes **`stuff1; stuff2; F`** to be executed.
c	T	F	No exception occurs. **`F`** is not executed.
d	T	F	Suppose exception type E1 is a subtype of E2 and an exception of type E1 occurs. **`stuff1; F`** is executed

e	T	F	C extends B, and B extends A. Object x is of type C. The value of "C instanceof A" is true.
f	T	F	If computing something takes expected time $O(n^2)$ for an input of length n , there could be specific inputs for which the function returns in time $O(1)$
g	T	F	$O(3n^4 + 2n + 7)$ is the same as $O(n^4)$
h	T	F	$O(\log (n^3))$ is the same as $O(\log n)$
i	T	F	Autoboxing occurs when a primitive type such as `int` is automatically translated to a value of the corresponding reference type, such as `Integer`.
j	T	F	A static initializer for a class is a block of code that will be executed when the class is loaded, but this won't occur until the first time the class is referenced.
k	T	F	Given a list of objects, Java's reflection features make it possible to identify the objects that define a **`void bark(int volume)`** method and, for those that do, to invoke **`bark(10)`** for a particular instance.
l	T	F	If Quicksort is used to sort a uniformly random vector with no repeats, the expected performance is $O(n \log n)$, but could be $O(n^2)$ if the pivot function always picks the smallest value in the vector.
m	T	F	Same, but now "*if the pivot always picks the median value in the vector*."
n	T	F	If class C extends B, and C overrides operation x, and b is of type B, then a method call to `b.x(args)` invokes first B's version of x, then C's version of x.
o	T	F	If you try to access the value of an unitialized Integer field in an object x of class A, a null pointer exception will be thrown.
p	T	F	Same, but "*the value will be 0*."

3 (22 points: 12 for correctness, 5 style, 5 efficiency). Finally, a coding question! Whew!

You are working for Google-Bike on a new Android cycling app. You are given an object `m` representing a `Map`: it has nodes (objects from a class `Node`) with directed edges (from class Edge). Each node has a field `neighbors` which is of type `LinkedList<Edge>`, and each edge has a field named `target`, giving the node to which it points, and other information such as distance, slope, difficulty, etc. `Edge` and `Node` objects don't have unique identifiers, although you can add fields that you need (see below).

<u>There can be multiple ways to get to the same node</u>, via different edges (like: Over the Mountain versus Around the Mountain).

Google Bike adds a field to the nodes in a `Map` to track routes. That is, each `Node` now has an extra vector called `next[r]`. For route `r` (an `integer` id) the value is either null (if the node isn't on route `r`) or it corresponds to one of the edges in the neighbors list. We'll consider cases in which there are two routes: Route 0 and Route 1. A route starts at some start node, and you follow it node by node until you get back to the start. The Map class is also extended: it now has a vector field `start[r]` giving, for route `r` the `Node` at which that route starts.

Write a method `boolean m.compareRoutes(int routeIdA, int routeIdB)` that is called on a Map object m specifying two route ID's, and that returns true if bike routes A and B are identical except for their start points and false otherwise. Although bike routes are loops that start and end at a designated start node, routes have no other loops. Assume that the route Id numbers are within range.

If you need to add additional fields to the `Node` or `Edge` class, you must tell us precisely which class the field will be added to, what its type and name are, and how it will be used. If the default value is non-null (or non-zero, for a base type like int), you must indicate precisely how it would be initialized.

Put your solution on the next page

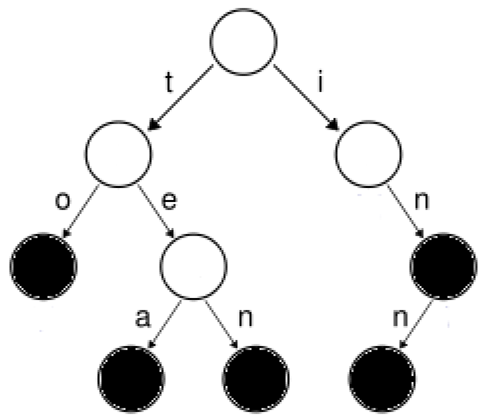
4. (20 points) Search Trees

(a) *(5 points)* Recall that an invariant is a property that always holds for a data structure. What is the invariant that distinguishes a binary search tree from an "ordinary" binary tree?

(b) *(5 points)* Imagine that we are implementing a delete method for a binary search tree. Deleting a value in a leaf node with no children is easy, just “delete” the node by setting the parent’s reference to it to null. However, deleting an internal node this way is not recommended as it will also disconnect its entire subtree. Instead, we swap the value at the internal node to be deleted with that of a leaf node, and then delete the leaf.

Which leaf would be a good candidate to swap the internal node’s value with so that the BST is still correct after the deletion?

(c) *(5 points)* A **trie** is a kind of tree data structure used to store a sorted set of strings. To look up a string, begin at the root and follow the path of characters in the string. Nodes store a boolean value to indicate that the path from the root to that node represents a complete string stored in the set, and not a partial prefix. This allows a word that is a substring of another to be stored, for example, "in" and "inn" shown below. The edges in our trie will be labeled with just a single character.



Trie containing "to", "tea", "ten", "inn"

Assuming a 26-character alphabet, what is the Big-O complexity of the lookup operation for a trie containing *n* strings of maximum length *m*? How does this differ from a binary search tree containing the same set of strings?

(d) *(5 points)* Imagine you are implementing a spellcheck feature for a word processor. Your program needs to be able to identify misspelled words, and also propose corrections. Would a trie be a good data structure for storing a dictionary? Explain.

5. (22 points) The Amazing Tales of the Oaksly Service

URL shortening websites like bit.ly work much like a hashtable where a cryptic short string (e.g., the XYZ in http://bit.ly/XYZ) is the key, and a long URL is the value. You decide to start a company for yet another URL shortening service and call it oaks.ly. Using your knowledge from CS 2110, you decide to maintain the maps from short URLs to long ones entirely in an in-memory hashtable just like the ones we used in class.

(a) (6 points) Your first employee, Gary, got his CS degree in a correspondence program. He is a bit mystified as to why we need to use a hashtable. He suggests maintaining a list of (short URL, long URL) pairs instead. What are the pros and cons of a hashtable versus a list, and why is a hashtable a good choice for Oaksly?

(b) (6 points) Gary is amazed by the idea of a hashtable, but can't believe they really exist. Briefly explain how hashtables are implemented. (3-5 sentences)

(c) *(5 points)* You're going to need a good hash function for your custom-built hashtable data structure. Gary learned that every character has a numerical value dictated by the ASCII encoding scheme (for example 'A' = 65), so he suggests using the sum of the ASCII values of the characters in short URL as the hash key. Is that a good idea? Explain briefly.

(d) *(5 points)* What is the *worst possible* performance for a hashtable? What hash code function could provoke that sort of worst lookup and store operation performance for Oaksly?

Extra Credit. (6 points) The Oaksly Service Part II: The Empire Strikes Back

Congratulations! Gary managed to debug his hashtable-version of the Oaksly service and the company is a great success; Wired magazine already proclaimed you the new King of the Internet. Everyone on the planet uses your service. Money is pouring in faster than you can spend it. But one day, Gary shows up looking pasty and sick. The Oaksly computer will run out of memory sometime next week and when it does, your new riches will be lost. He can't add more memory: the machine is maxed out!

You rush out to Best Buy and pick up a few hundred gigabytes of file space. But how will you use files as a "backing storage" area for the Oakly data? Don't write any code; just tell us how Gary should do it. (After all, you're the boss and he just works there! This is what founding a company is all about. Maybe you'll give him a raise if he doesn't mess up.)

To make things simple, assume that a file contains objects from a special kind of class that inherits from a class called "FileObject". When you access a "file object" the disk needs to be read (if you look at the value of the object) or written (if you change the object). But otherwise these file objects are like other objects. *Hint: File I/O is slow. A good solution <u>won't access the storage area more often than needed.</u>*