NAME: _____ NETID _____

CS2110 Fall 2014 Final

There are 4 questions, some with several parts. Check now that you have all 10 pages. Write your answers in the spaces provided. Use the back of the pages for workspace or longer answers. Ambiguous answers will be considered incorrect. The exam is closed book and closed notes. Do not begin until instructed.

We grade the final very carefully, knowing that you will not have a chance to look at the graded final until you come back in January. Grades will be posted when we have them ready; this may be several days.

You have 2.5 hours. Good luck! And have a nice winter break!

Start by writing your name and netid —legibly—at the top of each page.

	True/False	Crossword	Short Answers	Programming	Total
Max	20	22	30	28	100
Score					
Grader					

NAME: _____ NETID _____

1. **True-false questions (20 points)**

A. **T** The average-case and worst-case efficiency of binary search on a sorted array are equal.

B. **F** The average-case and worst-case efficiency of checking if an element is contained in a HashSet is O(1). [`false. the worst case on a set with n elements is O(n)`]

C. **F** Java has three kinds of variable: the field (instance variable), the static variable, and the local variable. [False. There is also the parameter]

D. **T** If method **equals** considers two objects equal, **hashCode** should return the same value for both objects but may not. [`true --if HashCode has been written properly`]

E. **F** In Java, hashCode() in class Object will return the same value every time you run the program. [False, depends on implementation / platform. Memory addresses are not repeatable]

F. **F** Graphs: Using an adjacency list representation, you can determine if two vertices are connected by an edge in O(1) time. [False. Is true for adjacency matrix]

G. **F** When building a hash set using linear or quadratic probing, method remove(x) can be implemented by setting the array element at the appropriate hashCode(x) to null. [False; this messes up the probing]

H. **F** Calling an Object's notify() method will notify all threads waiting on the Object's synchronization lock. [False, only one is notified, use NotifyAll to notify all]

I. **T** Super keyword is used to invoke overridden method which overrides one of its superclass methods. [True]

J. **F** The following language has an infinite number of sentences (i.e. an infinite number of sentences can be generated from nonterminal Sentence). Note: the nonterminals of this grammar are Sentence, S1, Noun, Verb, and Object

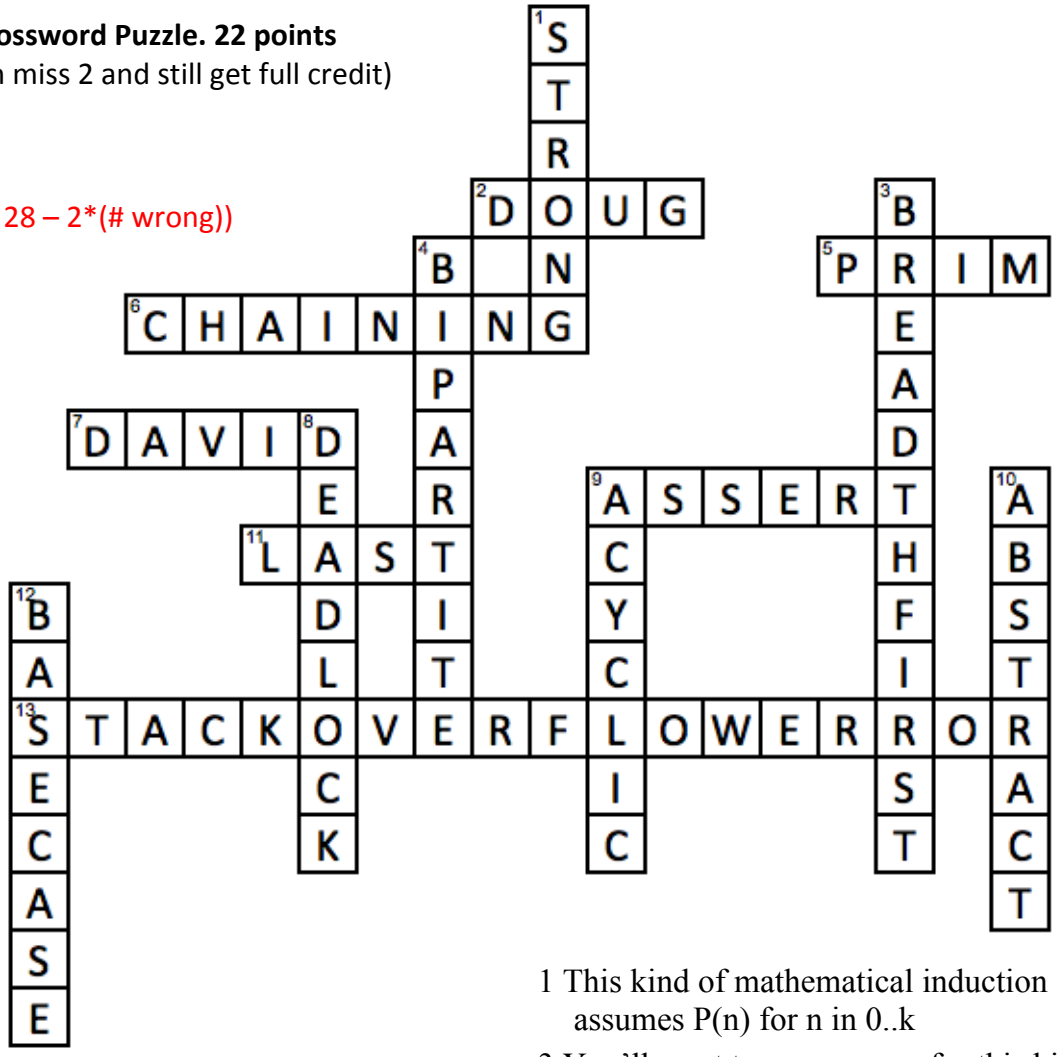
Sentence --> S1
Sentence --> Noun Verb Object
S1 --> S1 .
Noun --> dog | cat
Verb --> eats
Object --> food
[False, The rule S1 → S1 . thought infinite recursion, never generates a sentence]

NAME: _____ NETID _____

2110 Crossword Puzzle. 22 points
(you can miss 2 and still get full credit)

Grade:

Min(22, 28 – 2*(# wrong))



Across

2 First name of an instructor

5 Author of a CS2110 graph algorithm

6 Alternative to probing to solve collisions when hashing

7 First name of an instructor

9 Use this kind of statement to check preconditions of a method

11 A stack has a first-in-____-out behavior

13 Recursively calling the same function forever leads to this kind of Error

DOWN

1 This kind of mathematical induction assumes P(n) for n in 0..k

3 You'll want to use a queue for this kind of graph traversal

4 A 2-colorable graph is called this

8 When a group of threads are waiting to synchronize on an object that another group has already synchronized on, _____ may occur.

9 Topological sort works only if the directed graph is _____

10 Make a class this so that it can't be instantiated

12 Ends recursion and starts a proof by induction (two words)

NAME: _____ NETID _____

3. Short Answer Questions. (30 points)

3A. Topological Sort. (4 points) Below is pseudocode for topological sort. State below the condition under which the topological ordering is unique, i.e. there exists only one topological ordering.

```
/** Return a list of the nodes of graph g in topological order.
    Precondition: the graph has no cycles.  */
public List<Node> topologicalOrder(Set<Node> g) {
    Set<Node> g1  = a copy of g (so changing g1 will not change g);
    LinkedList<Node> res = new LinkedList<Node>();

    // Loop invariant: res contains the first res.size() nodes of a
    // topological sort of g1, in sorted order
    while (res.size() < g.size()) {
        Let n be a node of g1 with no incoming edges;
        res.add(n);
        Remove all edges from g1 that connect to n;
    }
    return res;
}
```

Short Answer: The sort is unique if at the first statement of the loop body ("Let n be a node of …") there is always exactly one node of g1 with no incoming edges.

3B. Exception Handling. (4 points) Execute the call **mm(0)** of the following method, writing the output generated by execution to the right.

```
public static int mm(int x) {
    try {
        System.out.println("one");
        int b= 5/x;
        System.out.println("two");
        return b;
    } catch (RuntimeException e) {
        System.out.println("three");
        int c= 5/(x+1);
        System.out.println("four");
    }
    System.out.println("five");
    int d= 5/x;
    System.out.println("six");
    return d;
}
```

WRITE YOUR OUTPUT HERE

one

three

four

six

NAME: _____ NETID _____

3C. Parsing. (5 points) Below is a simple context-free grammar (ε denotes the empty string):

S -> BS | ε
B -> α B β | B β B | α β

Identify which of the following strings are sentences of this grammar (i.e. can be derived from S) by writing YES or NO beside each one.

α α No

α No

β β α No

α β β α β α α α α β β β No

α β β α β Yes

3D. Connected Graphs. (4 points) A graph is said to be **biconnected** if two paths that do not share edges or intermediate vertices exist between every pair of vertices. Draw a picture of **two** different biconnected graphs, each with **four** vertices.

Any two of (Diamond graph, square graph, tetrahedral graph)

3E. Tree Traversal. (4 points) Although you can uniquely construct a binary tree from either its preorder and inorder traversals or its postorder and inorder traversals, more than one binary tree can have the same preorder traversal and the same postorder traversal. Draw two *different* binary trees (with unique node values) that have the *same* preorder and postorder traversals.

```
Pre-order [1,2] and post-order [1,2] could be either
```

NAME: _____ NETID _____

3F. Hashing with Collisions. (4 points) Consider search keys that are distinct integers. Suppose the table size is 7, the hash function is h(key)=key % 7 and linear probing resolves collisions. Where in the hash table do the following search keys appear after being added? 4, 6, 20, 14

0: 20
1: 21
2:
3:
4: 4
5:
6: 6

3G. Heaps. (5 points) Suppose we wanted to add another procedure **merge** to our class Heap. Procedure **merge** would add in all the elements from the argument heap. If we have two heaps, one with *n* nodes and the other with *m* nodes where $n < m$, what is the tightest bound worst-case time complexity for **merge**?

It will take O(m log m) time to take the nodes out of the m-sized list.

It will take O(m log(n+m)) time to put them in the originally n-sized list

So it will take O(m log(2m)), or O(m log(m)) time

NAME: _____ NETID _____

4. Programming. (28 points)
4A. Coding with Invariants (8 points)

Below is the skeleton of a program segment to "sort int array b" using selection sort. The loop invariant is given and MUST be used. Complete the initialization, loop condition, and loop body. You are given 2 points for providing code that ensures that each of the 4 loopy questions are answered correctly.

Remember: b.length could be 0, meaning that the array is empty.
Remember: Look carefully at the loop invariant!
Remember: You do not need to and should not write a nested loop of Java code; instead, write a high-level pseudocode statement that says what must be done.

```
// Sort int array b using selection sort
// put your Initialization here
k= b.length-1;      //worth 2 pts

// invariant P: b[0..k] <= b[k+1..] AND b[k+1..] is sorted

while  (   k >= 0  ) {  // Could also do k > 0   worth 2 pts
       Swap b[k] with the largest value in b[0..k] //worth 2 pts
       k= k-1;  //worth 2 pts
}
```

NAME: _____ NETID _____

4B. Bad Bits. (6 points) The following *is a flawed implementation* of a **set** of prime natural numbers. In this question, you will revise method **add(n)**.

```
/** A set implementation that adds natural numbers ONLY if they are prime. */
class PrimeSet {
    // List-backed set implementation:
    private ArrayList<Integer> set = new ArrayList<Integer>();

    /** Add n to the set ONLY if its a prime number and ignore otherwise.
      * Precondition: n>0. */
    public void add(int n) {
        // Return if not prime:
        for (int k=2; k < n; k++) {
                if (n%k == 0) return;
        }

        set.add(n);
    }
}
```

Below, rewrite **add(int n)** so that it is (1) correct and (2) thread-safe while supporting many parallel calls to **add(int n)**. Do not allocate any other class member fields. Do not write an improved prime number test.

```
// Note: making add "synchronized" is heavy-handed and reduces
// parallelism:
public void add(int x) {
      // (No synchronization req'd for prime checking)
      // Return if not prime:
      for (int k=2; k < x; k++) { if (x%k == 0) return; }

      // (Operations on "set" need to be synchronized)
      // Ignore if already in "set" list:
      synchronized(set) {
            if (!set.contains(x)) set.add(x);
}
```

NAME: _____ NETID _____

4C. Isomorphic Trees. (8 points) Class TreeNode is defined as:

```
public class TreeNode<T> {
        public TreeNode left, right; // Left and Right nodes, or null if none
        public T    value;
}
```

Write the following method body:

```
/** Return true iff the trees rooted at a and b have the same tree
 *  structure (this has nothing to do with field value). */
public boolean isIsomorphic(TreeNode a, TreeNode b) {
boolean A=   a == null;
boolean B=   b == null;

if (A && B) return true;   // both null
if (A || B)) return false; // only one is null

// both nonnull → recurse
return isIsomorphic(a.left, b.left) && isIsomorphic(a.right, b.right);
}
```

NAME: _____ NETID _____

4D. Concurrent Bakery. (6 points) Consider the following code that describes a scenario where you have one producer thread (a baker) and ten consumer threads (ten hungry customers):

```
class Bakery {
    int nLoaves = 0; // Current number of waiting loaves
    final int K = 10; // Shelf capacity

    public synchronized void produce() {
        if (nLoaves == K) {
            this.wait(); // Wait until not full
        }
        ++nLoaves;
        this.notifyall(); // Signal: shelf not empty
    }

    public synchronized void consume() {
        if (nLoaves == 0) {
           this.wait(); // Wait until not empty
        }
        --nLoaves;
        this.notifyall(); // Signal: shelf not full
    }
}
```

Either (1) Explain why this code is thread-safe or (2) explain why it may not work and describe a fix for it.

It is thread-safe, since only one process may be referencing **this** object at a time. But it may not work. Suppose a consumer finds nLoaves == 0. It waits until it is notified that nLoaves > 0. But ALL consumers who have been waiting will do the same thing, since all are notified. The first K to get through will find a loaf and take it. But consumer K+1 will assume there is a loaf of bread and do ++nLoaves, this "taking" a loaf, even if there is none there. Solve this by making the consume if-statement into a while-statement.