Name: NetID:

Prelim 2

CS 2110, November 20, 2014, 7:30 PM

	1	**2**	**3**	**4**	**5**	**Extra**	**Total**
Question	True/False	Short Answer	Complexity Induction	Trees	Graphs	Extra Credit	
Max	20	10	15	25	30	5	100
Score							
Grader							

The exam is closed book and closed notes. Do not begin until instructed.

You have **90 minutes**. Good luck!

Write your name and Cornell **netid** at the top of EACH page! There are 5 questions on 12 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your booklet is still stapled together. If not, please use our stapler to reattach all your pages!

We have scrap paper available. If you do a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam, so that we can make sense of what you handed in.

Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to fit your answers easily into the space provided.

In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that its good style.

1. True / False (20 points)

a)	T	F	If a method calls a procedure that requires $O(n \log n)$ operations once and calls another procedure that requires n operations 150 times, the complexity of the method is $O(n \log n)$.
b)	T	F	The tightest bound on worst-case complexity of removing an item from a heap containing n items is $O(\log n)$.
c)	T	F	Since `Integer` is a subclass of `Object`, `LinkedList<Integer>` is a subtype of `LinkedList<Object>`
d)	T	F	A `HashMap` has worst case lookup cost $O(1)$.
e)	T	F	No comparison-based sorting algorithm can achieve best-case complexity better than $O(n \log n)$.
f)	T	F	A `hashCode()` method must return a prime integer larger than 7.
g)	T	F	On the same graph, Prim's and Kruskal's algorithm will not necessarily return the exact same minimum spanning tree.
h)	T	F	A "`for (T elem: container)`" statement can be used to iterate over the members of container if container is an array.
i)	T	F	A method that takes $O(\log n)$ will always compute a result faster than an $O(n)$ method, assuming n has the same value for each method.
j)	T	F	An inner class can access private instance members of the surrounding class.
k)	T	F	It will always take $O(\log n)$ time to find an arbitrary node in a binary search tree of size n.
l)	T	F	If you don't provide a different hash function, inserting a `String` of length l into a `HashSet` of size n takes $O(l)$ time
m)	T	F	If an undirected connected graph has n nodes, the minimum spanning tree will have $n-1$ edges and no cycles.
n)	T	F	If a graph is planar, then it is always 4 colorable.
o)	T	F	On a connected unweighted graph, Depth First Search is guaranteed to find the shortest path to any given node from a defined start node.
p)	T	F	Depth-first search maintains a queue of nodes to be visited.
q)	T	F	In a valid coloring of a graph, only non-adjacent vertices can have the same color.
r)	T	F	On an unweighted graph Depth First Search will visit all vertices at distance d from the start node before any vertex at distance $d+1$.
s)	T	F	When a graph is stored as an adjacency-matrix, checking if there is an edge between two vertices takes $O(1)$ time in the worst case.
t)	T	F	The tightest bound on the worst-case complexity of inserting an item into a heap containing n items is $O(1)$.

2. Short Answer (10 points)

(a) 2 points If a heap is implemented in a Java array, what is the index of the parent of the node at index k, assuming $k \neq 0$?

(b) 2 points Suppose you want to write an exception class and then throw an instance of it when a heap is empty. Below, write down the heading for this exception class. You get to name the class and what, if anything, it extends or implements.

(c) 3 points If a binary tree has n nodes, what is the maximum and minimum depth it could be?

(d) 3 points Draw a non-planar graph of minimal size.

3. Complexity and Induction (15 points)

(a) 3 points What is the tightest bound on the time complexity of the following function? **Be careful! This requires you to know the complexity of all operations and function calls below.**

```
/** Return a new ArrayList that is seq in reverse */
public static<T> ArrayList<T> reverseArrayList(ArrayList<T> seq) {
    ArrayList<Integer> newList = new ArrayList<Integer>(seq.size());
    for (T i : seq) {
        newList.add(0, i);
    }
    return newList;
}
```

(b) 4 points Can the function body in part (a) be rewritten to improve the time complexity —perhaps inserting in a different order? If so, rewrite it below and state what the best possible time complexity is. If not, explain why no improvement over your answer from part (a) is possible.

(c) 8 points Let $P(n)$ be: $8^n - 5^n$ is divisible by 3. Prove by induction that $P(n)$ holds for all integers $n \geq 0$.

(i) **2 points** Write the base case(s) for your proof.

(ii) **3 points** Write the inductive hypothesis for your proof.

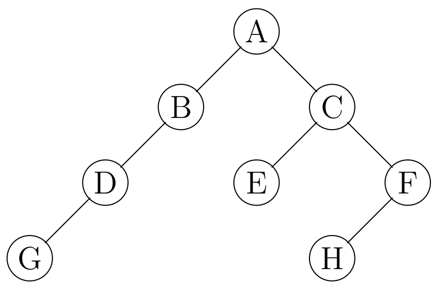
(iii) **3 points** Complete the inductive step for your proof.

4. Trees (25 points)

(a) 3 points Given the following pre-order and in-order tree traversals, draw the tree.

In-order	B, D, C, E, A, F
Pre-order	C, B, D, E, F, A

(b) 3 points Given the following tree, state the post-order tree traversal.



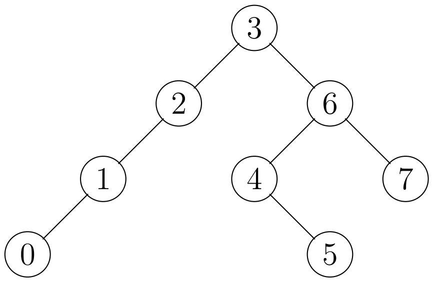
(c) 19 points The following questions pertain to class `Node` defined below:

```
public class Node {
    public int value;
    public Node left;
    public Node right;
    public int nodes; // The number of nodes in this tree, including this node
}
```

(i) **7 points** Complete method `initNumNodes` according to its specification:

```
/** Initialize field nodes in t and in all of its descendants.
  * Return the number of nodes in t. */
public static int initNumNodes(Node t) {







}
```

(ii) **12 points** The inorder traversal of a tree visits the nodes in a certain order. An example is given in the tree below, where the number in each node is the node's inorder-index. Note that if tree `t` has 3 nodes in its left subtree, its root is numbered 3. *Note: The values of each node may be different from the numbers given below. The tree is not necessarily a Binary Search Tree.*

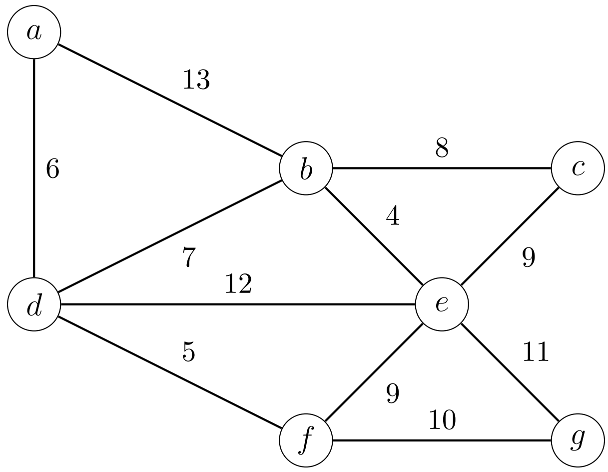


Complete method `isNodeN` according to its specification. There is no need to check the preconditions; assume they are true. You may and should use field `nodes` from the previous question. **Your solution must run in time $O(h)$ for full credit, where h is the height of the tree.**

```
/** Return true if and only if node number n of the inorder
  * traversal of tree t is nth
  * Precondition: t != null && nth != null && n >= 0 && n < t.nodes
  * and field nodes is initialized in t and all its descendants. */
public static boolean isNodeN(Node t, int n, Node nth) {





}
```

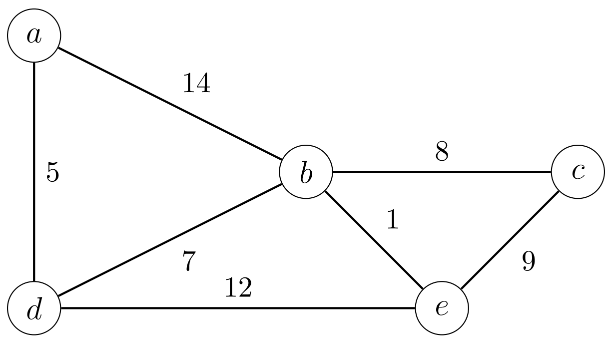
5. Graphs (30 points)

(a) 5 points Consider the following graph:



List the edges of the graph above in the order they would be added to a minimum spanning tree by Kruskal's algorithm. If a starting node is required, start with node a. An edge should be listed by the pair of vertices it joins. For example, the edge with weight 12 above would be edge (d, e)

(b) 10 points Consider the following graph:



Execute Dijkstra's shortest-path algorithm on the graph above with a as the start node. We show the initial state, before iteration 0 of the main loop, in the second column below, giving the frontier set (the blue set in the description of Dijkstra's algorithm given in lecture), and the L value for each node so far.

For each iteration 0, 1, 2, ..., first write in the appropriate place in the table the Selected node, that is, the node that gets removed from the frontier set. Below that, write the new value of the frontier set and the L values that change on this iteration.

Node label	Distance						
Selected Node	N/A						N/A
Frontier Set	$\{a\}$						
$L[a]$	0						
$L[b]$	∞						
$L[c]$	∞						
$L[d]$	∞						
$L[e]$	∞						
Iteration	Init	0	1	2	3	4	Final

(c) 15 points Consider the following class representing a vertex in an undirected graph.

```
public class Vertex {
    public HashSet<Vertex> neighbors;
}
```

Let v be a vertex of a graph g. The connected component of g that contains v is the subgraph contaning all vertices (and their edges) that are connected to v by some path. g could consist of several connected components, none of which are connected to each other.

Complete functions `hasOne` and `dfs` below according to their specifications. We have filled in part of the body of the first method to help you out.

```
/** Return true iff the set of vertices given by g consists of exactly one
    connected component. */
public static boolean hasOne(ArrayList<Vertex> g) {
    HashSet<Vertex> visited= new HashSet<Vertex>();
    // Put declarations of local variables and initialization here



    for (Vertex v : g) {




    }



}

/** Precondition: v is not in visited. Add to visited every Node reachable
    from v along paths of vertices that are not in visited. */
public static void dfs(Vertex v, HashSet<Vertex> visited) {










}
```

Name: NetID:

Extra Credit (5 points, all or nothing)

Consider a situation where your input int array of size n is almost sorted.

[1, 4, 3, 2, 5, 7, 6, 8]

In this array, each number is at most k positions away from its correctly sorted position. In the above example, $k = 2$. Complete procedure `sort` below according to its specification. You may use whatever data structure you would like and may use its methods without implementation as long as your intentions are clear

Your solution must run in $O(n \log k)$ time and be clearly organized for credit.

```
/** Sort the array.
  * Precondition: Each element is no more than k
  * positions away from where it should be in sorted order */
public void sort(int[] b, int k) {






}
```