Name:__________________________________

Question	Points
I.	/12
II.	/30
III.	/10
IV.	/12
V.	/12
VI.	/12
VII.	/12
TOTAL:	/100

Instructions

1. This is a closed-book, closed-notes exam.
2. You have 120 minutes for the exam.
3. Calculators are not allowed.
4. Show all of your work.
5. Clearly indicate your final answer.

Definitions

The following definitions are copied verbatim from the textbook:

Definition 2.1.
$T(N) = O(f(N))$ if there are positive *constants* c and n_0 such that $T(N) \leq cf(N)$ when $N \geq n_0$.

Definition 2.2.
$T(N) = \Omega(g(N))$ if there are positive *constants* c and n_0 such that $T(N) \geq cg(N)$ when $N \geq n_0$.

Definition 2.3.
$T(N) = \Theta(h(N))$ if and only if $T(N) = O(h(N))$ and $T(N) = \Omega(h(N))$.

I. True/False (1 point each, 12 points total)

For each question in this section, indicate whether the statement is TRUE or FALSE. Circle **ONE** answer. Choose the **BEST** answer.

Note: In the questions below, if two running times have equivalent rates of growth, then neither one is considered asymptotically faster than the other. An operation has to be "much faster" than another operation to be considered asymptotically faster. For example, n is not asymptotically faster than $2n$, but n is asymptotically faster than $n \log n$.

1. The function $n \log n + n$ is $\Omega(n^3)$.

   TRUE FALSE

2. The function $3n^3 - 2n^2$ is $\Theta(n^3)$.

   TRUE FALSE

3. The function $n^2 + 4n$ is $O(n^3)$.

   TRUE FALSE

4. In the worst case, insertion into a red-black tree is asymptotically faster than insertion into a binary heap.

   TRUE FALSE

5. In the worst case, insertion into a binary heap is asymptotically faster than insertion into a sorted linked list.

   TRUE FALSE

6. In the average case, insertion into a hash table (using a "good" hash function) is asymptotically faster than insertion into a red-black tree.

   TRUE FALSE

7. In the worst case, searching for a key in a red-black tree is asymptotically faster than searching in a sorted linked list.

   TRUE FALSE

8. In the worst case, searching for a key in a binary heap is asymptotically faster than searching in a sorted linked list.

   TRUE FALSE

9. In the average case, searching for a key in a hash table that uses a “good” hash function is asymptotically faster than searching in a red-black tree.

   TRUE FALSE

10. In the worst case, removing the minimum item from a red-black tree is asymptotically faster than removing the minimum item from an unsorted linked list.

   TRUE FALSE

11. In the worst case, removing the minimum item from a red-black tree is asymptotically faster than removing the minimum item from a min heap.

   TRUE FALSE

12. In the average case, removing the minimum item from a red-black tree is asymptotically faster than removing the minimum item from a hash table that uses a “good” hash function.

   TRUE FALSE

II. Short Answers (30 points total)

1. Order the following functions by growth rate. Indicate which functions grow at the same rate.

   $n^2, \quad \log(n^2), \quad n^3, \quad n\log(n^2), \quad n^2\log(n^2)$

2. Argue mathematically that the function $3n^2+4n-5$ is $O(n^2)$. Justify your answer by providing the constants c and n_0 from the definition of $O(n^2)$ and showing that these constants satisfy the definition.

3. (3 points) In a red-black tree, can we ever have a node where the size of its left subtree (i.e., the number of items in the left subtree) is more than **five times** the size of its right subtree? Explain.

4. (3 points) Explain how a threaded program could run faster than an equivalent unthreaded program on the same machine. In what situations might a threaded program be slower than the unthreaded version?

5. (3 points) What is the purpose of a lock in a threaded program?

6. (3 points) Suppose we use the division method for hashing into a hash table with 13 slots. That is, key k is hashed into slot number k % 13. Show the placement of the keys:

   4, 29, 20, 42, 35, 16

   in the hash table using **linear probing** for collision resolution. *Show your work.*

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

7. (3 points) Suppose we use the division method for hashing into a hash table with 13 slots. That is, key k is hashed into slot number k % 13. Show the placement of the keys

   4, 29, 20, 42, 35, 16

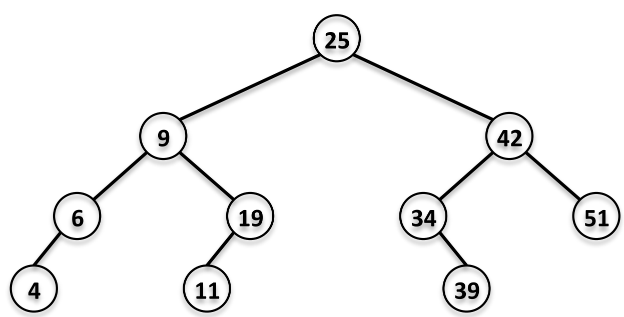
   in the hash table using **quadratic probing** for collision resolution. *Show your work.*

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

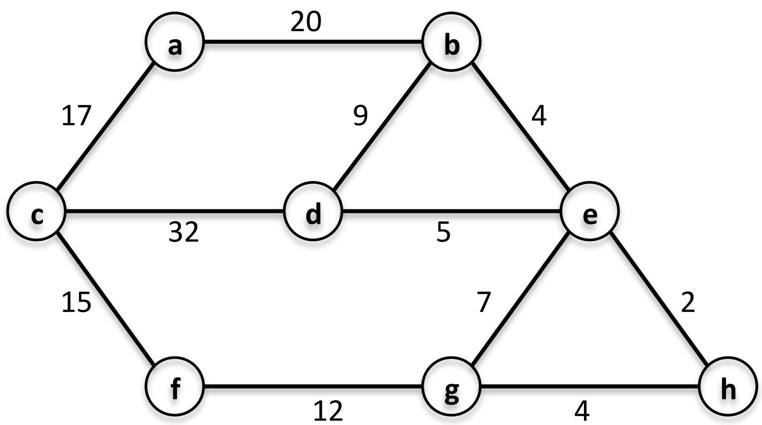
8. (3 points) The following array represents a disjoint set union data structure. Draw the corresponding diagram for this array. (Use the diagrams shown in the figures in the textbook and during lecture.)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
5	11	11	-1	9	8	4	11	-1	4	-1	10	-1	5	11

9. (3 points) Consider the following binary search tree. Draw the resulting binary search tree after inserting 7, inserting 8 and deleting 9. Make sure that you insert and remove the items in that order.



10. (3 points) Consider the following undirected graph. List the first 5 edges in the order chosen by Kruskal's algorithm during its construction of a minimum spanning tree. Include only the edges that will be included in the minimum spanning tree. You may specify an edge by specifying the nodes that are its endpoints (e.g, (a, b)).



Edge #1 = ____________________

Edge #2 = ____________________

Edge #3 = ____________________

Edge #4 = ____________________

Edge #5 = ____________________

III. Proof by Induction (10 points)

Let T be a rooted binary tree. Each node in T may have 0, 1 or 2 children. We define the degree of a node x to be the number of children of x.

Prove by induction that the number of nodes in T is 1 plus the sum of the degrees of the nodes in T.

You must prove this by induction. For partial credit, you must attempt a proof by induction. *Hint:* induct on the number of nodes in the tree. *Hint, hint:* remove the root and consider some cases.

IV. Short Program 1 (12 points)

Consider the following declarations for a singly-linked list data structure that uses a dummy header.

```
public class Node {
   int data ;
   Node next ;
}

public class SingleLL {
   Node header ;
   ...
}
```

Write a Java method for the `SingleLL` class with the following signature:

```
public void keepThird() ;
```

The `keepThird()` method should keep every third item in host linked list and should remove the other items. For example, if the host linked list initially held 7, 1, 9, 11, 14, 8, 5, 6, 4, 2 then after the call to `keepThird()` the host linked list should hold 9, 8, 4. You should not assume that any other methods are implemented for the `SingleLL` class — i.e., you should provide all the code to implement `keepThird()`.

For full credit, your method should run in $O(n)$ time where n is the number of items in the host linked list.

V. Short Program 2 (12 points)

Write a Java method that works with the `LinkedList` class from the Java Collections API. Your method should have the following signature:

```
public static boolean unique(LinkedList<Integer> aList) ;
```

The `unique()` method assumes that `aList` holds a sorted list of `Integer` values. It looks through the items in `aList` and determines whether every item is unique. For example, if `aList` held 4, 9, 11, 12, 15, then `unique()` should return `true` because each value in the list appears only once in the list. On the other hand, if `aList` held 4, 9, 11, 11, 12, 15 then `unique()` should return `false`.

For full credit, your implementation of the method should run in $O(n)$ time where n is the number of items in `aList`.

VI. Short Program 3 (12 points)

Write a recursive method `predecessor()` in Java, for a binary search tree of `Integer` values which, given a key k, returns the largest item with key value strictly less than k. If no such value exists, then `predecessor()` returns `null`. The value k may or may not be in the binary search tree. You may assume that the tree does not have duplicate key values.

For example, suppose the binary search tree held the keys: 2, 4, 5, 7, 8, 11, 15, 17, 19, 21. Then, the call `predecessor(19)` should return 17. The call `predecessor(10)` should return 8 and `predecessor(1)` should return `null`.

Your method must be recursive. For full credit, your method must run in time proportional to the height of the tree. If you use a reasonable declaration of the binary search tree data structure, then you do not have to provide the declaration.

VII. Short Program 4 (12 points)

Implement an operation in Java, called `deleteSS()`, for a binary minimum heap of `int` values. Here, the heap is stored in an array of `int` as usual. The `deleteSS()` method removes and returns the second smallest item from the heap. The `deleteSS()` method must fix up the heap so that none of the heap properties are violated. (This must be done by `deleteSS()` and not by calling another method.) You may assume that the heap has at least 4 items. For full credit, your method must run in $O(\log n)$ time.