Name: _____

Email address: _____

Quiz Section: _____

CSE 326 Winter 2010: Midterm Exam

(closed book, closed notes, calculators o.k.)

Instructions Read the directions for each question carefully before answering. We will give partial credit based on the work you **write down**, so show your work! Use only the data structures and algorithms we have discussed in class or that were mentioned in the book so far.

Note: For questions where you are drawing pictures, please circle your final answer for any credit.

Good Luck!

Total: 108 points. Time: 50 minutes.

Question	Max Points	Score
1	15	
2	6	
3	15	
4	15	
5	6	
6	6	
7	10	
8	10	
9	15	
10	10	
Total	108	

1. (15 pts) **Big-O, Big Ω, Big Θ True/ False**

Indicate for each of the statements below whether the statement is true or false. You do not need to state the reason, but a reason may help you get partial credit.

TRUE / FALSE a) $100\ N^3 = \Omega(N^2)$

TRUE / FALSE b) $2^{N/2} = O(2^N)$

TRUE / FALSE c) $N^2 + N^2 = O(N^3)$

TRUE / FALSE d) $N \log N + N^2 = \Theta(N^2)$

TRUE / FALSE e) $\frac{1}{2}(N^2) = \Theta(N^3)$

2. (6 pts) **Recurrence Relationships -**
Please circle your answer. ***Be sure to keep track of constants exactly*** (e.g. don't use "C" in your answer). We want the actual function, not just the big-O class.

Suppose that the running time of an algorithm satisfies the recurrence relationship

$$T(N) = 2 * T(N/2) + 2 \text{ for N and integer greater than 1.}$$

and

$$T(1) = 5.$$

Solve for T(N). In other words express T(N) as a function of N. For partial credit, please show your work.

3. (15 pts) **Binary Min Heaps. To avoid ambiguity, for this question please give your answer in Java.** Recall that a binary min heap with *n* elements can be stored in an array A, where A[1] contains the root of the tree. Given that values are stored in A[1] to A[size], `percolateUp` and `percolateDown` are defined below:

```
// Given that values are stored in A[1] to A[size],
// percolate the value A[i] up as needed.
void percolateUp(int[] A, int size, int i);

// percolate the value A[i] down as needed.
void percolateDown(int[] A, int size, int i);
```

a) Implement the `deleteMin` method (as described in class) below. You may call `percolateUp` and `percolateDown` as needed:

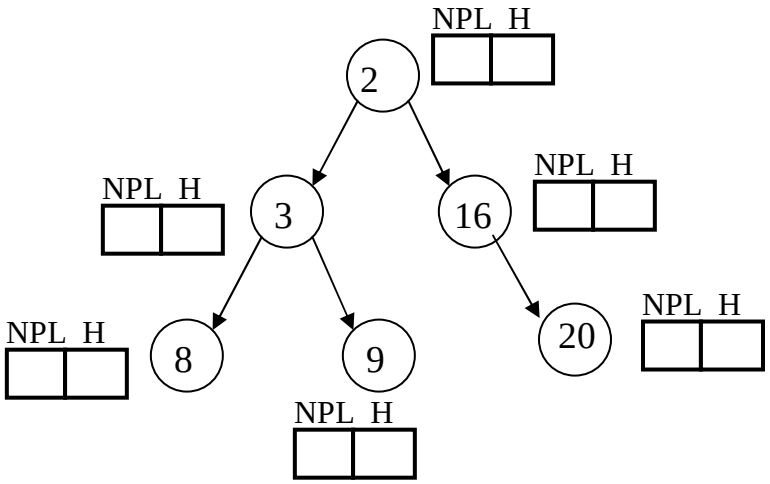
```
// Given that values are stored in A[1] to A[size],
// remove and return the smallest value in the heap.
int deleteMin(int[] A, int size) {
```

b) Implement the `percolateUp` method below:

```
// Given that values are stored in A[1] to A[size],
// percolate the value A[i] up as needed.
void percolateUp(int[] A, int size, int i) {
```

4. (15 pts) **Trees**

a.) (6 pts) Mark the following properties for each node of the tree below in the space indicated for each node: **Null Path Length (NPL)** and **Height (H).**



b.) (9 pts) Also, circle **yes** or **no** to indicate whether the tree above might represent each of the following data structures. If you circle **no,** give **one specific reason** why the tree could **not** be that data structure.

- AVL tree yes no
- Binary Min heap yes no
- Leftist heap yes no

5. (6 pts) **Running Time Analysis**

Give the best O-bound on the *worst case* running time for each of the following in terms of *N*. **No explanation is required**, but an explanation may help for partial credit. Assume that all keys are distinct.

a) Merging two skew heaps, the largest heap is of size N

b) Inserting a value into a leftist heap of size N

c) Merging two binary heaps, the largest heap is of size N

6. (6 pts) **Short Answer**

a) How long would you expect inserting a value into a binomial queue to take *on average*? ***Justify your answer***.

b) ***Give one reason*** why inserting a value into a d-heap containing N items might be faster than inserting a value into a binary heap of the same size.

7. (10 pts) **AVL Trees**

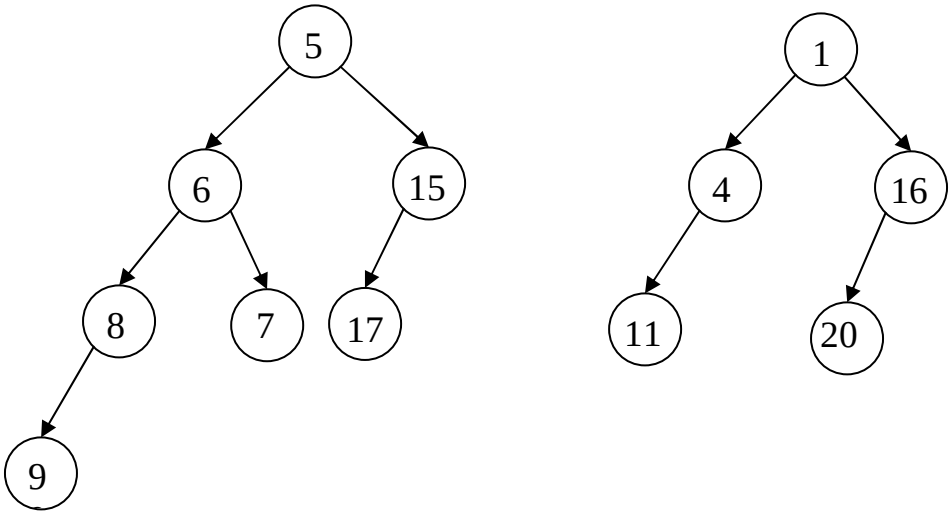
a) Draw the AVL tree that results from inserting the keys 4, 1, 2, 3, 9, 5, 7, 15 in that order into an initially empty AVL tree. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***

b) Give a Post-order traversal of your final tree here: _____

7. b) Don’t’ forget to give a post-order traversal of your final tree from part a)! *Write your answer at the bottom of the previous page.*

8. (10 pts) **Leftist Heap Merge**

Merge the following two leftist min heaps using the leftist heap merge described in class. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.*** You may continue your answer to this question on the next page if needed.



8. **Leftist Heap Merge** continued

9. (15 pts) **Binary Min Heaps**

a) (10 pts) Draw the binary min heap that results from inserting 12, 1, 3, 7, 4, 5, 15, 0, 6 in that order into an **initially empty binary heap**. You do not need to show the array representation of the heap. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***

9. **Binary Min Heaps (**continued**)**

b) (5 pts) Draw the result of doing 2 deletemins on the heap you created in part a. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***

10. (10 pts) **Binomial Queues** –

Draw a binomial queue that could result from inserting 12, 1, 3, 7, 4, 5, 15, 0, 6 in that order into an **initially empty binomial queue**. You are only required to show the final queue, although if you draw intermediate queues, ***please circle your final result for ANY credit.*** You may continue onto the next page if needed.

10. **Binomial Queues** (continued)