Name: _____ KEY _____

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

$T(N) = 2 * T(N/2) + 2$ 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.

$$T(N) = 2 \cdot T\left(\frac{N}{2}\right) + 2$$

$$= 2\left[2 \cdot T\left(\frac{N}{4}\right) + 2\right] + 2$$

$$= 2\left[2\left[2 \cdot T\left(\frac{N}{8}\right) + 2\right] + 2\right] + 2$$

$$= 2^3 \cdot T\left(\frac{N}{8}\right) + 8 + 4 + 2$$

$$= 2^3 \cdot T\left(\frac{N}{8}\right) + 14$$

$$= 2^k \cdot T\left(\frac{N}{2^k}\right) + 2^{k+1} - 2$$

Want $\frac{N}{2^k} = 1$, so $N = 2^k$ also true:

$$T(N) = N \cdot T(1) + 2 \cdot N - 2$$

$$= N \cdot 5 + 2 \cdot N - 2$$

$$= 7N - 2$$

$$\boxed{T(N) = 7N - 2}$$

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) {

   if ( size < 1 ) throw new IllegalArgumentException();
   // or otherwise test for the case of an empty heap.

   int temp = A[1];
   A[1] = A[size];
   // size--;    // not required because size is a
   // parameter, has no effect.  Should have been
   // specified as a class method…
   percolateDown(A, size, 1);
   return temp;

   }
```

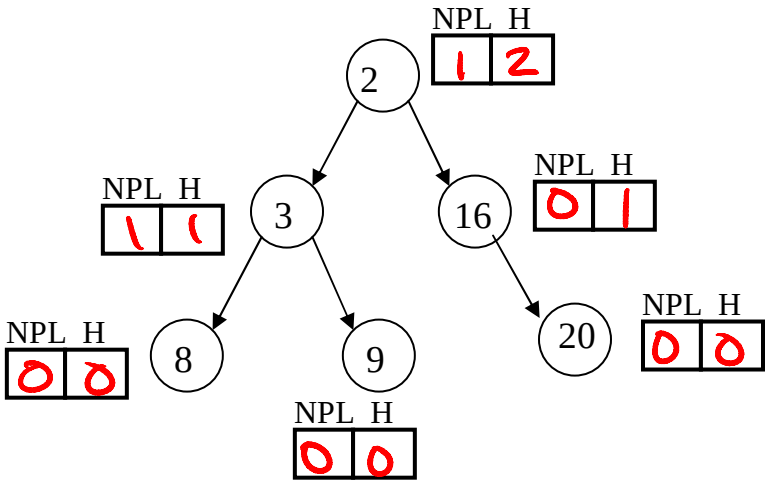
b) Implement the `percolateUp` method below:

```
// 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) {


   int temp;
   for (; (i > 1) && (A[i/2] > A[i]); i /= 2) {
        temp = A[1/2];
        A[i/2] = A[i];
        A[i] = temp;
   }
}
```

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**

Not ordered like a BST, 8 should not be to the left of 8 (or 2).

- Binary Min heap yes **no**

Not a complete tree, node 20 would need to be the left child of 16.

- Leftist heap yes **no**

Not leftist at node 16, the NPL of 16s left child is -1 which is < 0.

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

O(N) - Try merging a linked list 1,2, …N with a single value N+1

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

O(log N) – a merge of a single value with another heap, cost of merge is O(logN).

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

O(N) – use Floyds buildheap, copy all values into an array and then run buildheap.

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***.

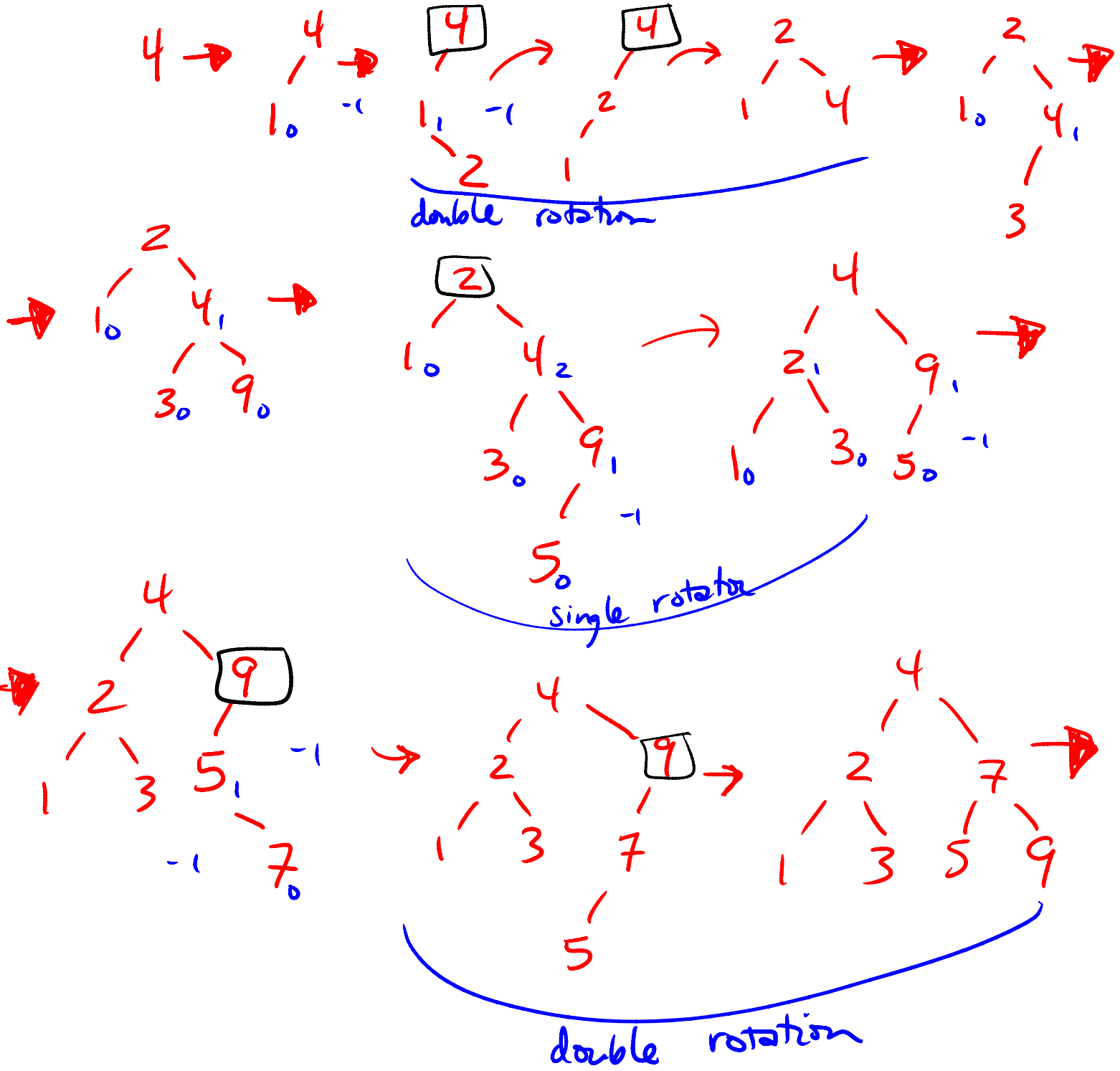
O(1) – Your worst case is like adding 1 to 111111. You must continually merge two trees of the same size together and carry them on to the next place, but then you already have a tree of that size so must merge it with the next one, etc. causing you to examine and merge with all trees in the BQ – O(log N). For the average case, we want to know when we might expect to see a 0 when moving from right to left, because when we find a 0, we can stop – we can leave any carry in that location. Assuming that our BQ size N is equally likely to be an odd number as it is an even number, we expect the BQ to have nothing in the B_0 location 50% of the time. Similarly, we expect nothing in the B_1 location 50% of the time. Thus, you expect to have to look at 2 locations on average before you find a B_k that does not exist.

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.

If d > 2 then the d-heap is likely to have fewer levels than a binary heap of the same size. On insertions you percolate up, and only compare with one parent. So you don't get to take advantage of the spatial locality of examining siblings that are next to each other. (Percolate down has to examine all siblings which is more than in a binary heap if d > 2, but hopefully the heap is shorter, and you do get to take advantage of locality in that case.)

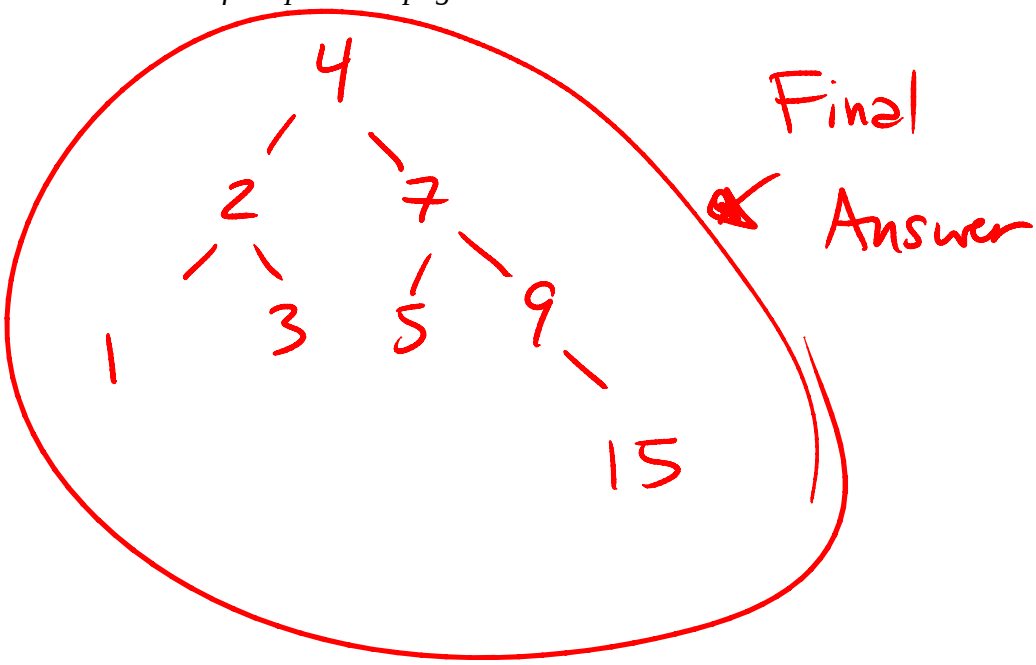
7. (10 pts) **AVL Trees**

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.***



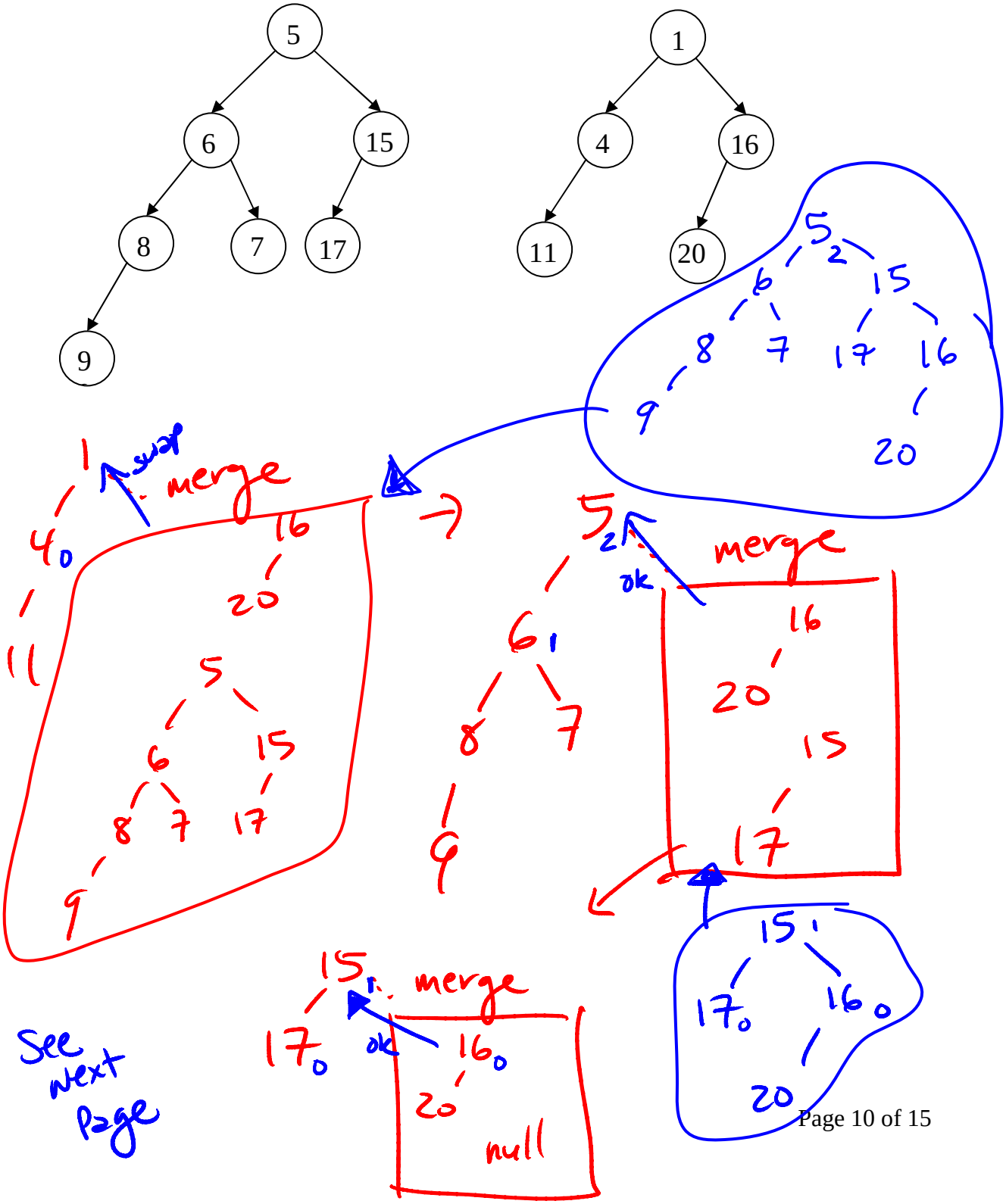
b) Give a Post-order traversal of your final tree here: _____**1, 3, 2, 5, 15, 9, 7, 4** _____

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.*

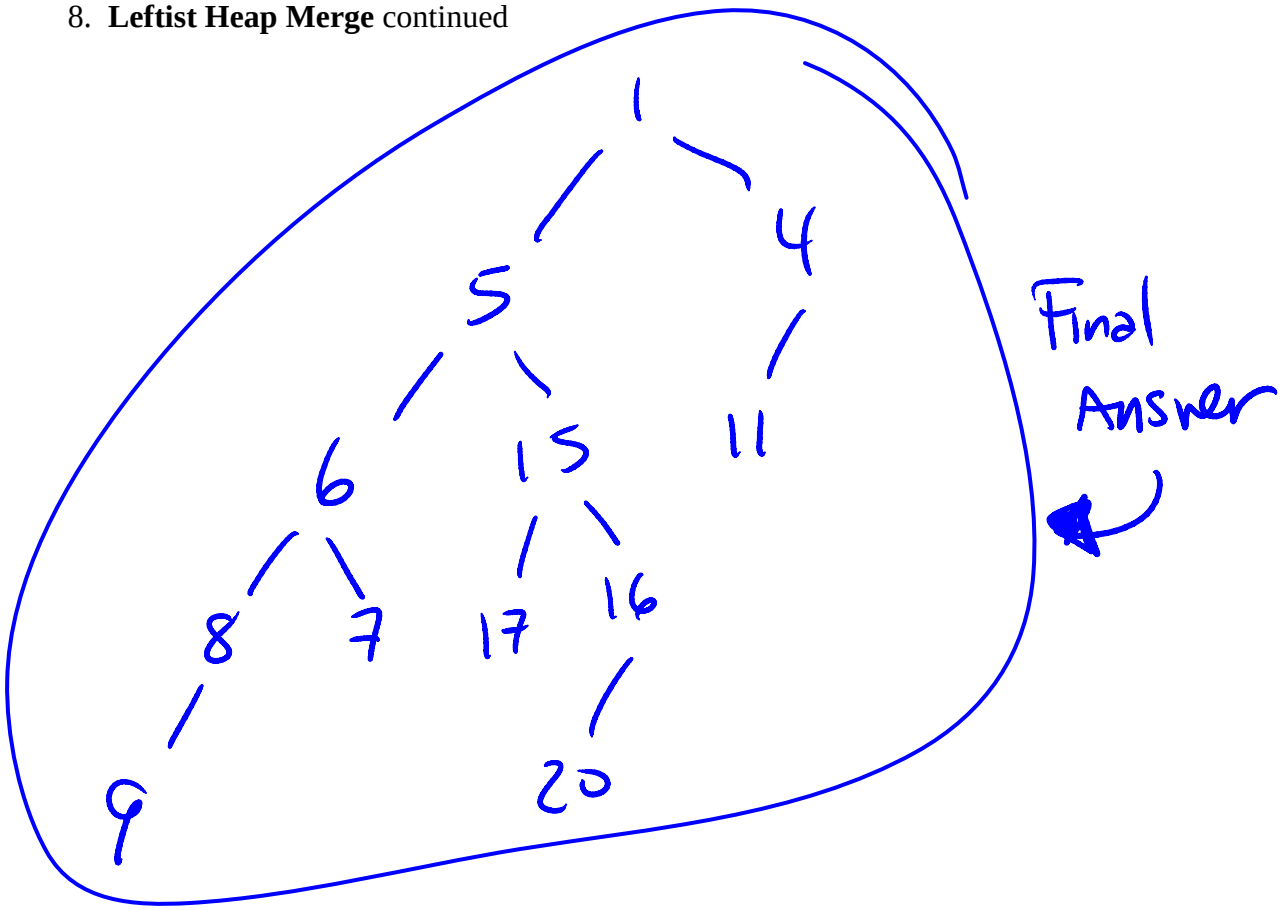


8. (10 pts) **Leftist Heap Merge**

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.

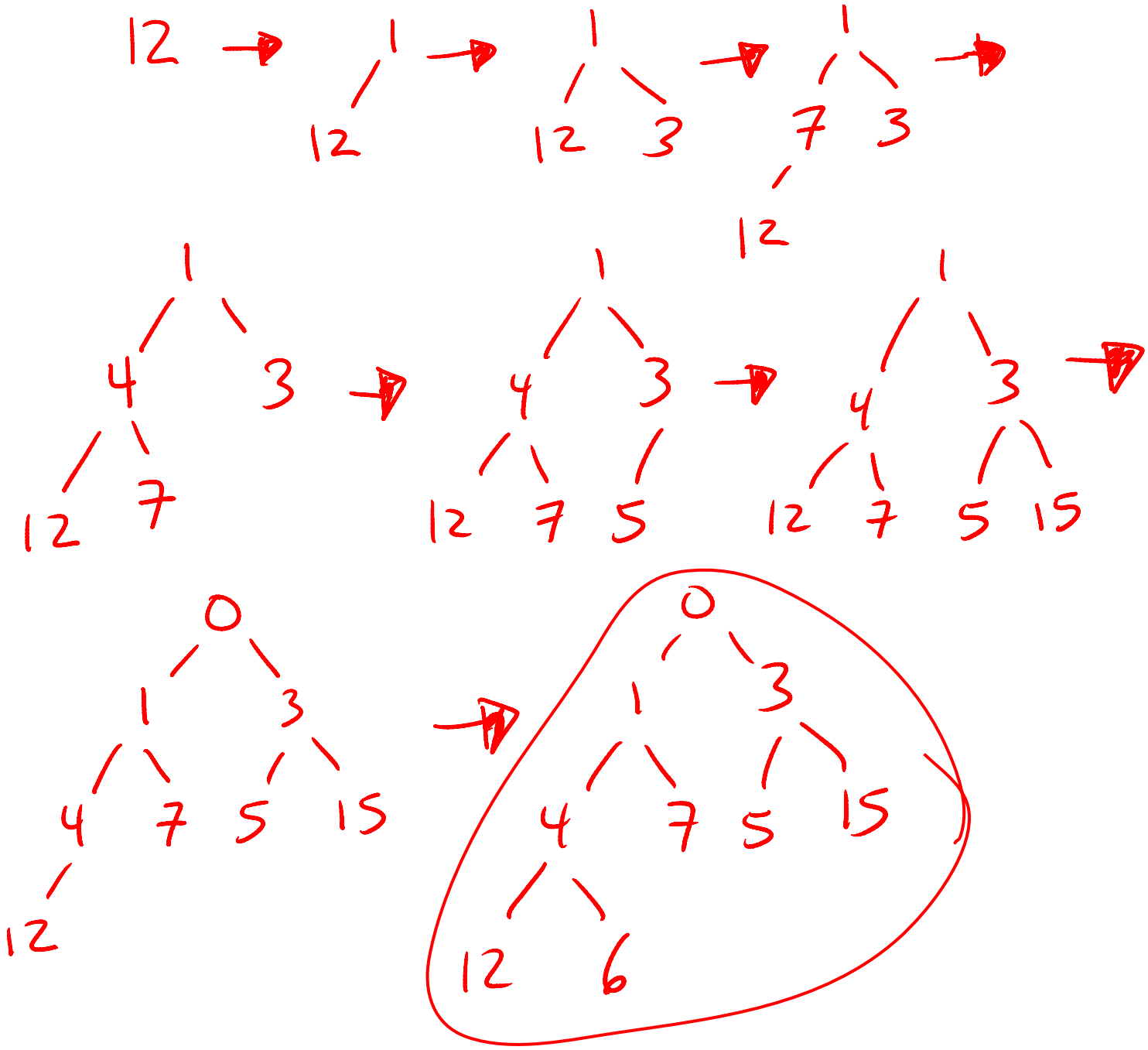


8. **Leftist Heap Merge** continued



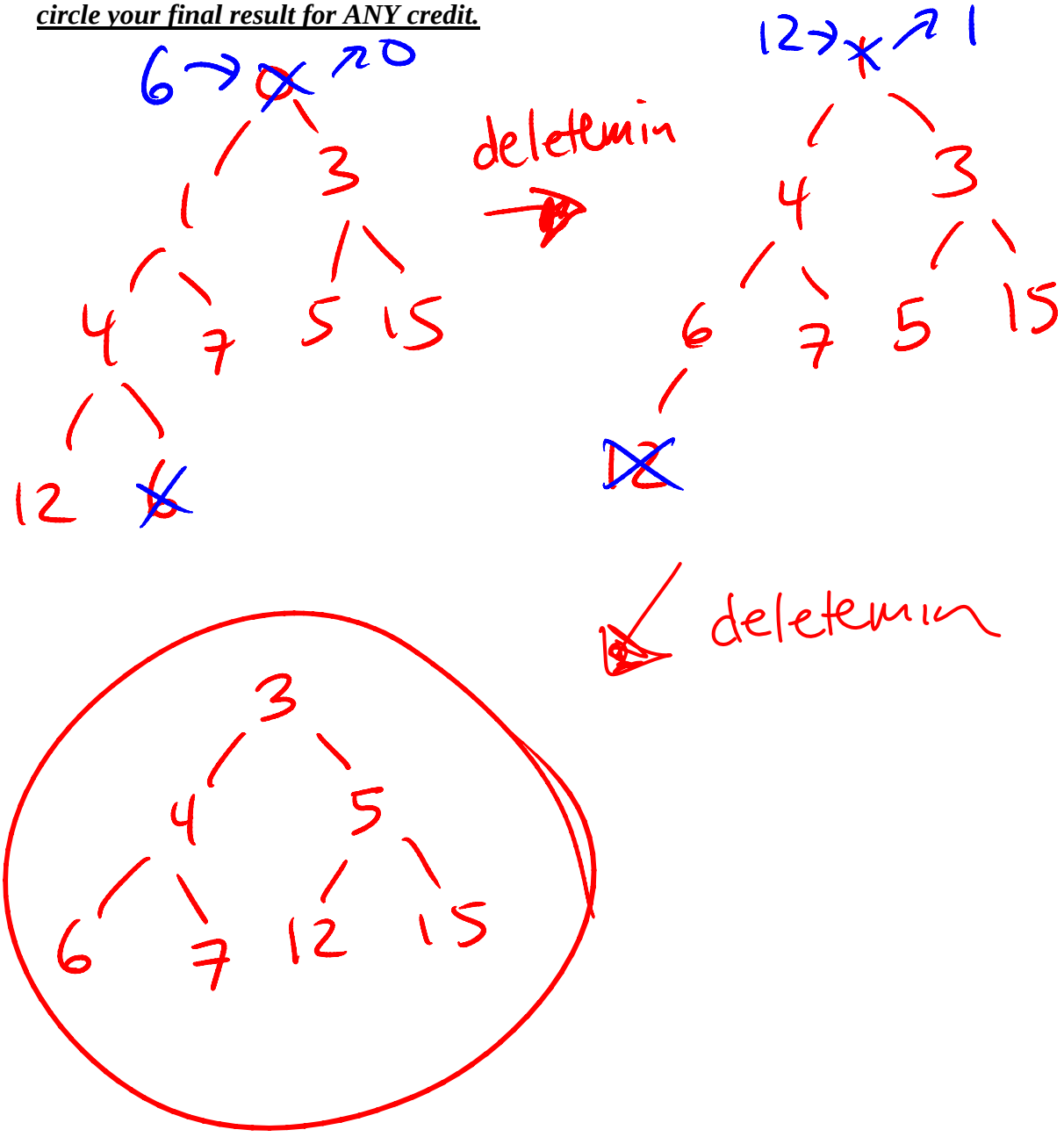
9. (15 pts) **Binary Min Heaps**

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.***



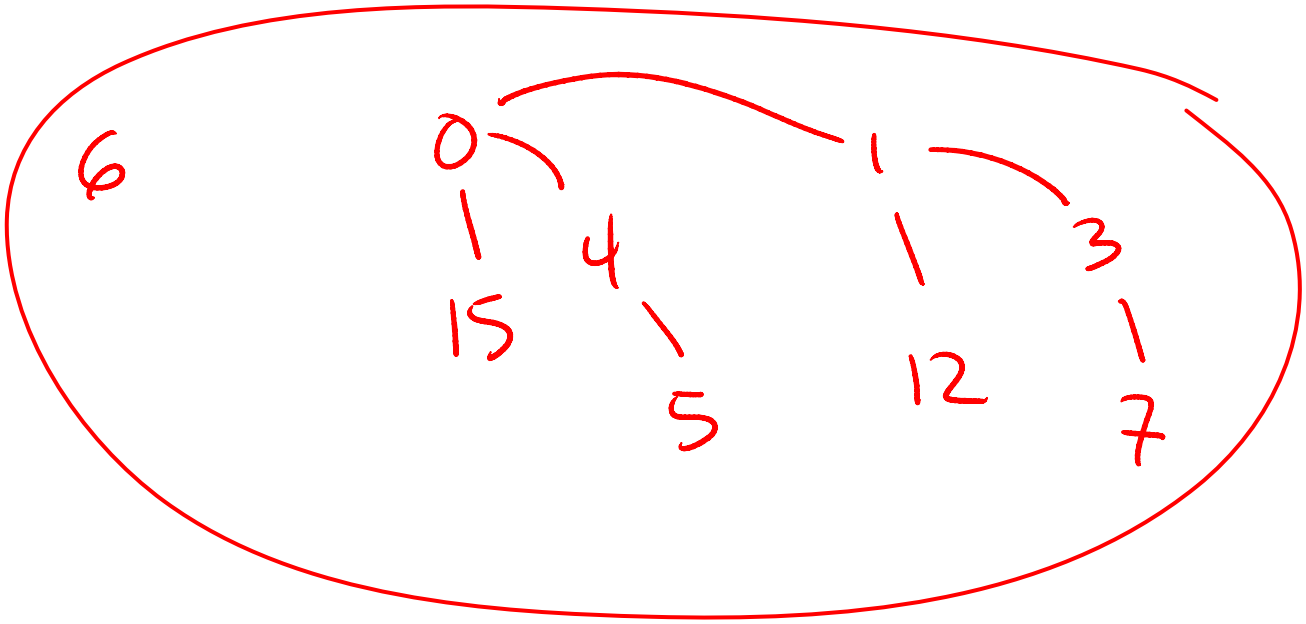
9. **Binary Min Heaps (**continued**)**

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



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)