Name: _____**Sample Solution**_____

Email address: _____

Quiz Section: _____

CSE 332 Winter 2013: Midterm Exam
(closed book, closed notes, no calculators)

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: 100 points. Time: 50 minutes.

Question	Max Points	Score
1	10	
2	12	
3	10	
4	6	
5	8	
6	8	
7	8	
8	14	
9	14	
10	10	
Total	100	

1. (10 pts) Big-Oh

(2 pts each) For each of the functions $f(N)$ given below, indicate the tightest bound possible (in other words, giving $O(2^N)$ as the answer to every question is not likely to result in many points). Unless otherwise specified, all logs are base 2. **You MUST choose your answer from the following** (not given in any particular order), each of which could be re-used (could be the answer for more than one of a) – e)):

$O(N^2)$, $O(N^{1/2})$, $O(N^3 \log N)$, $O(N \log N)$, $O(N \log \log N)$, $O(N)$, $O(N^2 \log N)$, $O(N^5)$, $O(\log N)$, $O(1)$, $O(\log^2 N)$, $O(N^4)$, $O(N^N)$, $O(N^6)$, $O(N \log^2 N)$, $O(2^N)$, $O(\log^4 N)$, $O(N^3)$

You do not need to explain your answer.

a) $f(N) = 50 \log N^2 + 100 N^2 \log N$

$O(N^2 \log N)$

b) $f(N) = (N^2)^3 + N^5$

$O(N^6)$

c) $f(N) = \log_4(N^4)$

$O(\log N)$

d) $f(N) = N \log^2 N + N \log \log N$

$O(N \log^2 N)$

e) $f(N) = N^{1/2} + \log N$

$O(N^{\frac{1}{2}})$

2. (12 pts) Big-Oh and Run Time Analysis: Describe the worst case running time of the following pseudocode functions in Big-Oh notation in terms of the variable n. Give an answer in the most simplified form (similar to options to choose from in Problem 1). *Showing your work is not required*

Runtime:

```
I.  void sunny (int n, int sum) {
        for (int i = 0; i < n * 1000; ++i)
            for (int j = 0; j < i; ++j)
                for (int k = n; k > 0; k--)
                    sum++;
    }
```

$O(n^3)$

```
II.  int happy (int n, int sum) {
        if (n < 10)
            return n/3;
        else if (n < 100)
            return happy (n/2, sum);
        else {
            for (int i = 0; i < n * n; ++i)
                sum++;
            return happy (n - 2, sum);
        }
    }
```

$O(n^3)$

```
III.  void smiley (int n, int sum) {
         for (int k = 0; k < n; ++k) {
             if (n < 100) {
                 for (int i = 0; i < k; ++i)
                     sum++;
             }
             for (int j = n; j > 0; j--)
                 sum++;
         }
      }
```

$O(n^2)$

```
IV.  void funny (int n, int sum) {
        for (int k = n; k > 0; k = k/2) {
            for (int j = n; j > 0; j--)
                sum++;
            for (int i = 0; i < n; i++)
                sum++;
        }
    }
```

$O(n \log n)$

3. (10 pts) Big-O, Big Ω, Big Θ

(2 pts each) For parts (a) – (c) circle whether the statement is true or false. Part (d) has its own instructions. You do not need to show your work for parts (a) – (c).

TRUE / ~~FALSE~~   a) $N^2 + N = \Theta(N^3)$ *Not* Ω

~~TRUE~~ / FALSE   b) $N^2 + N^3 = \Omega(N^3)$

~~TRUE~~ / FALSE   c) $100\,N^2 + 20N + 500 = \Theta(N^2)$

d) (4 points) Demonstrate that: $25n + 300$ is $O(n)$ by finding positive integers c and n_0 such that the definition of Big Oh is satisfied. You do not need to prove that these constants satisfy the definition, just specify the constants.

$$25n + 300 \leq C \cdot n \quad \text{for } n \geq n_0$$

Many possible values for C & n_0   *e.g.* $C = 325$

$n_0 = 1$

4. (6 pts) Recurrence Relationships -

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

$$T(1) = 10.$$

and

$$T(N) = T(N/2) + 3 \quad \text{for integers } N > 1$$

Find the closed form for T(N) *and show your work step by step*. In other words express T(N) as a function of N. Your answer should *not* be in Big-Oh notation – show the relevant *exact* constants in your answer (e.g. don't use "C" in your answer).

$$T(N) = T\left(\frac{N}{2}\right) + 3$$
$$= 3 + \left(3 + T\left(\frac{N}{4}\right)\right)$$
$$= 3 + 3 + \left(3 + T\left(\frac{N}{8}\right)\right)$$
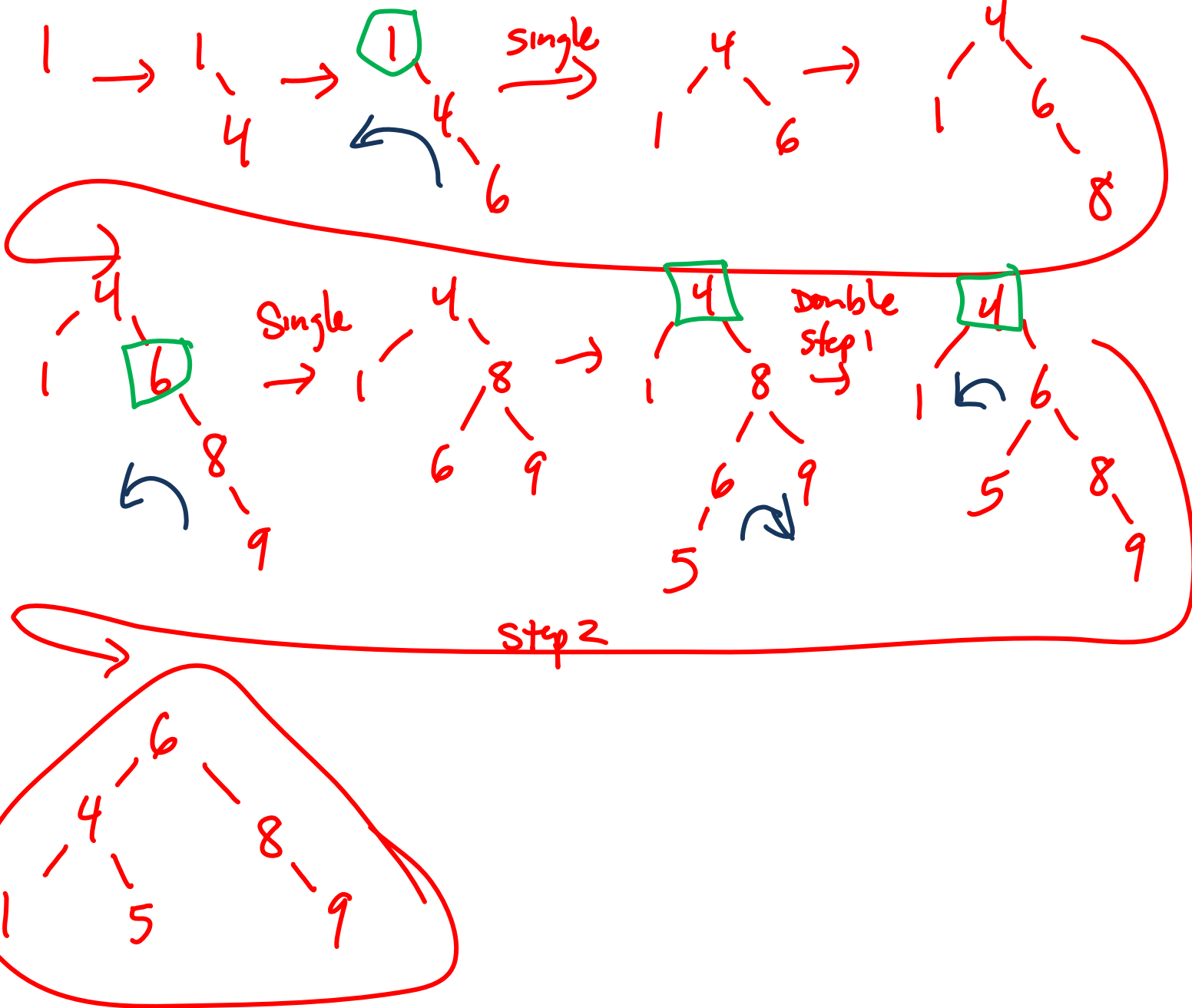$$= 3K + T\left(\frac{N}{2^k}\right)$$
$$T(N) = 3 \cdot \log_2 N + T(1)$$
$$T(N) = 3 \cdot \log_2 N + 10$$

Want:
$$1 = \frac{N}{2^K}$$
$$2^k = N$$
$$k = \log_2 N$$

5. (8 pts) Draw the AVL tree that results from <u>inserting the keys 1, 4, 6, 8, 9, 5, in that order</u> into an <u>initially empty AVL tree</u>. You are only required to show the final tree, although if you draw intermediate trees, ***please circle your final result for ANY credit.***

1 → 1 → (1) → Single → 4 → 4
     \   \   / \   / \
      4   4   1   6   1   6
                 \   \
                  6   8

→ 4 → Single → 4 → 4 → Double Step 1 → 4
   / \   / \   / \   / \
  1   6   1   8 1   8   1   6
       \   / \   / \   / \
        8   6   9   6   9   5   8
       / \   \   \
      ... 9   5   9

Step 2 →

          6
         / \
        4   8
       / \   \
      1   5   9

6. (8 pts) Trees

a) (4 pts) What is the minimum and maximum number of nodes in an **Binary min heap of height 6?**   (Hint: the height of a tree consisting of a single node is 0) *Give an exact number* for both of your answers – not a formula.

Minimum = $2^6 = 64$

Maximum = $2^{6+1} - 1 = 2^7 - 1 = 127$

b) (4 pts) What is the minimum and maximum number of nodes in an **AVL tree of height 5?**   (Hint: the height of a tree consisting of a single node is 0) *Give an exact number* for both of your answers – not a formula.

Minimum = 20

Maximum = $2^{5+1} - 1 = 2^6 - 1 = 63$

$$S(h) = S(h-1) + S(h-2) + 1$$
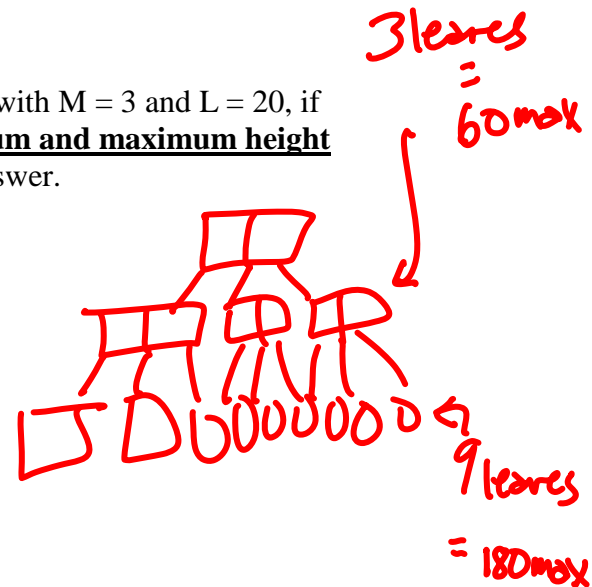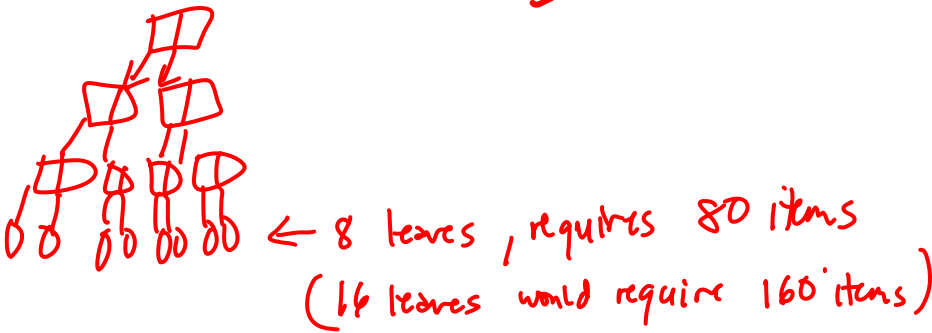$$S(5) = S(4) + S(3) + 1$$
$$= 12 + 7 + 1 = 20$$

h	nodes
0	1
1	2
2	4
3	7
4	12
5	20

7. (8 pts) B-trees

a) (4 pts) Given a B-tree (as defined in lecture and in Weiss) with M = 3 and L = 20, if you have inserted 100 items into the tree, what is **the minimum and maximum height of the tree**? Give an exact number, not a formula for your answer.

Minimum height = 2

Maximum height = 3

3 leaves
=
60 max

9 leaves
= 180 max

← 8 leaves, requires 80 items
(16 leaves would require 160 items)

b) (4 pts) Given the following parameters for a B-tree with M= 37 and L = 5:

     Key Size = 6 bytes

     Pointer Size = 8 bytes

     Data Size = 100 bytes per record (*includes* the key)

Assuming that M and L were chosen appropriately, **what is the likely size of a disk block** on the machine where this implementation will be deployed? Give a numeric answer and a short justification based on one or more equations using the parameter values above.

Disk Block $\geq$ L $\cdot$ 100

    = 500

Disk Block $\geq$ 500

Disk Block $\geq$ M $\cdot$ 8 + (M-1) 6

    = 8 $\cdot$ 37 + 6 $\cdot$ 37 - 6

    = 14 $\cdot$ 37 - 6

    = 518 - 6

    = 512

Disk Block $\geq$ 512 ← bigger of the two vals

Disk Block must be $\geq$ 512 if M+L were chosen wisely
probably = 512   since that is a power of 2.

Probably 512 bytes

8. (14 pts) Hash Tables
For a) and b) below, insert the following elements in this order: 80, 9, 8, 18, 29, 49. For each table, TableSize = 10, and you should use the primary hash function $h(k) = k \% 10$. If an item cannot be inserted into the table, please indicate this and continue inserting the remaining values.

a) Linear probing hash table:

0	80	18_2 29_1 49_1
1	18_3	29_2 49_2
2	29_3	49_3
3	49_4	
4		
5		
6		
7		
8	8	18_0
9	9	18_1 29_0 49_0

b) Quadratic probing hash table

0	80	29_1 49_1
1		
2	18_2	
3	29_2	49_2
4		
5	49_4	
6		
7		
8	8	18_0 49_3
9	9	18_1 29_0 49_0

The following two questions are about hashing generally, NOT about the two tables above.

c) What is the big-O worst case running time for an **insert** operation in a **linear probing hash table** containing N elements? (Assume there is empty space in the table.)

$$O(N)$$

d) What is the big-O worst case running time for a **find** operation in a **separate chaining hash table** containing N elements **where each bucket points to a sorted linked list**?

$$O(N)$$

9. (14 pts) Binary Min Heaps

a) (8 pts) Draw the binary min heap that results from <u>inserting 9, 4, 2, 6, 3, 7, 8, 5, 1 in</u> <u>that order</u> 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.***
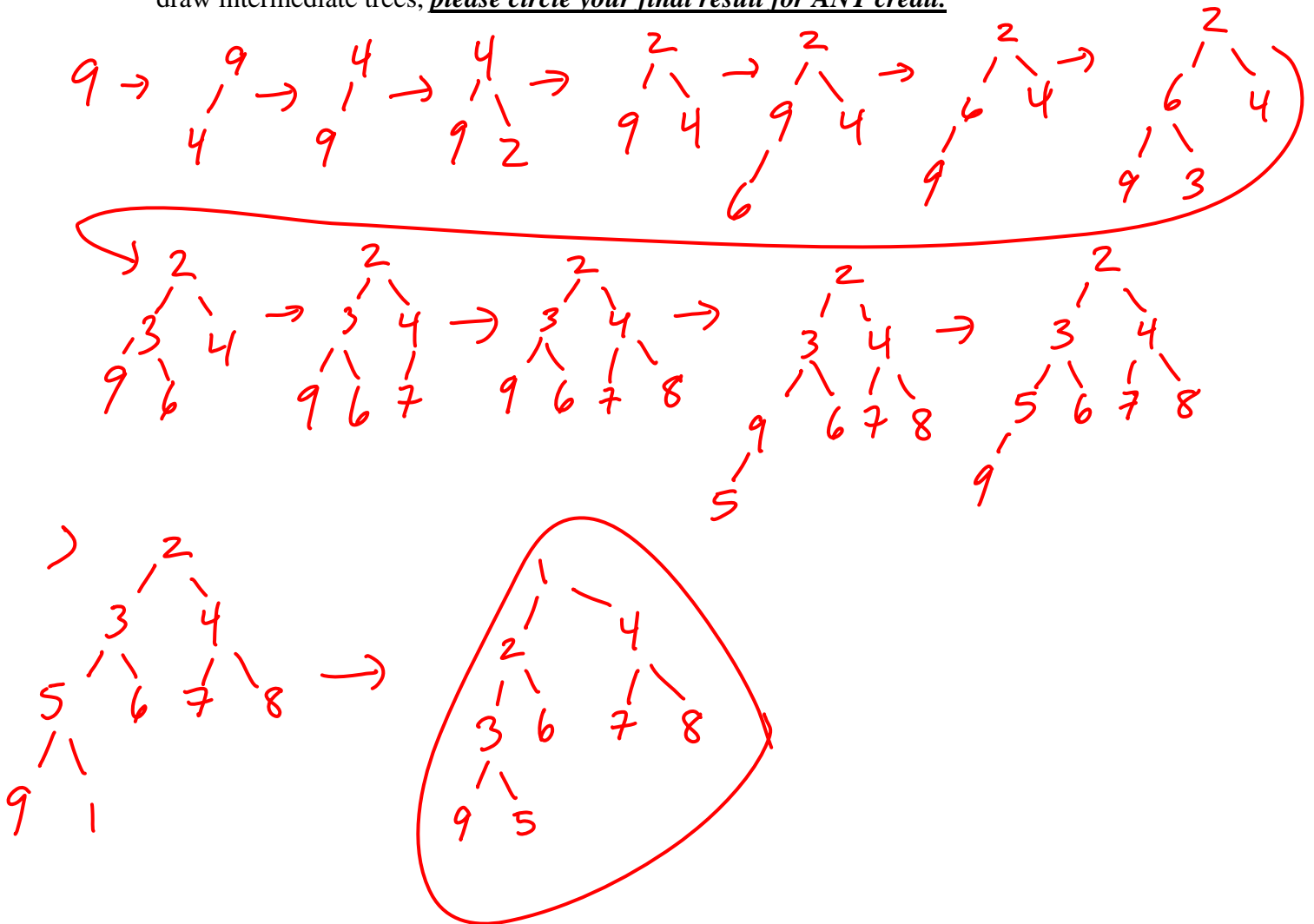
9. Binary Min Heaps (continued)

b) (4 pts) Draw the result of <u>doing 1 deletemin</u> 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.***
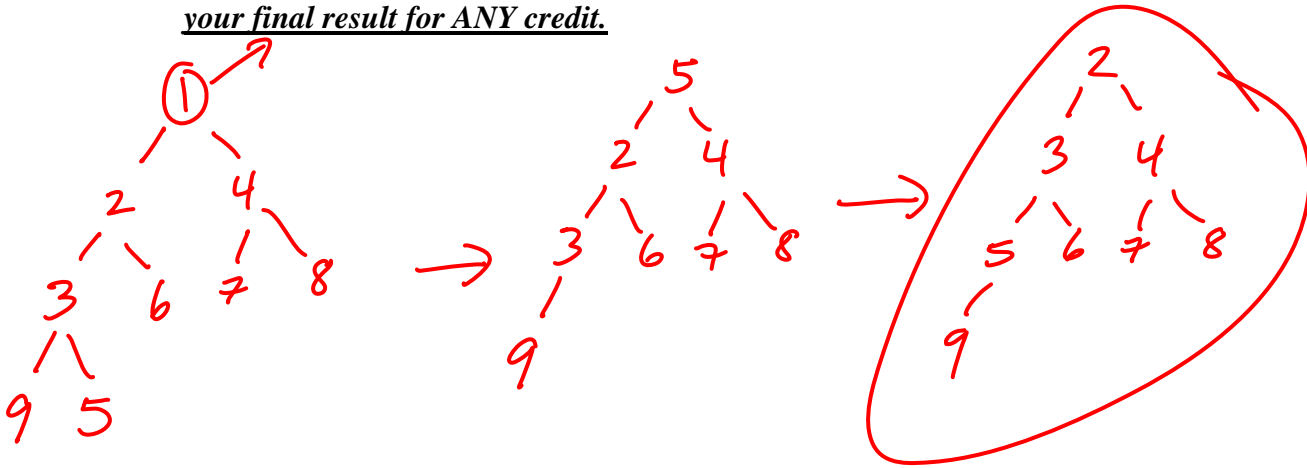

c) (2 pts) What is the big-O worst case running time for a **findmin** operation in binary min heap containing N elements?

$$O(1)$$

10. (10 pts) B-tree Insertion and Deletion

a) (2pts) In the B-Tree shown below, please write in the appropriate values for the interior nodes.

b) (4 pts) Starting with the B-tree shown below, insert **9**. Draw and circle the resulting tree (*including values for interior nodes*) below. Use the method for insertion described in lecture.
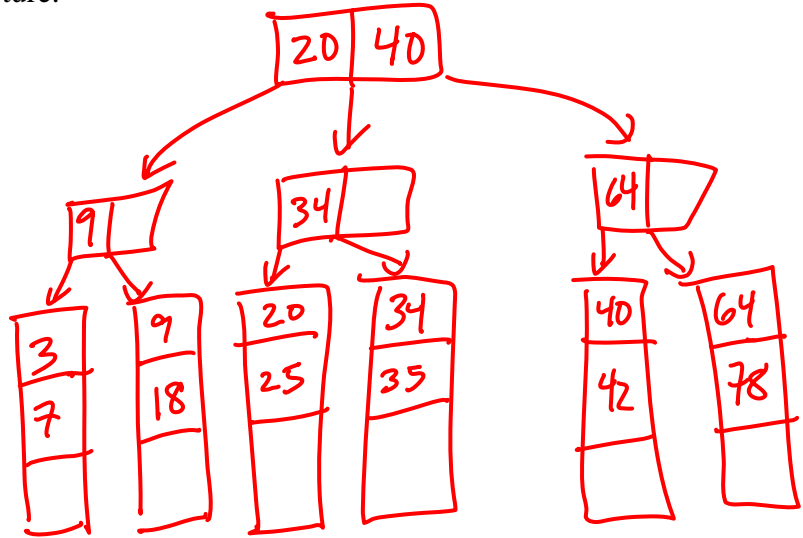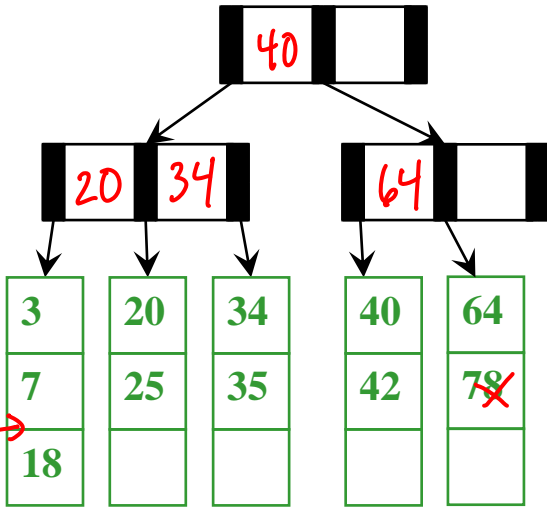
c) (4 pts) Starting with the **_original_** B-tree shown above on the left (**_before_ inserting 9**), delete **78**. Draw and circle the resulting tree (*including values for interior nodes*) below. Use the method for deletion described in lecture.