Name: Sample Solution

Email address: _____

Quiz Section: _____

CSE 332 Autumn 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	18	
2	6	
3	10	
4	6	
5	8	
6	6	
7	8	
8	15	
9	15	
10	8	
Total	100	

1. (18 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. **Your answer should be as "tight" and "simple" as possible.** For questions that ask about running time of operations, assume that the most efficient implementation is used.

You do not need to explain your answer.

a) *Find in a **separate chaining hash table** containing N elements where each bucket points to a sorted linked list (worst case)* O(N)

b) *Preorder traversal on an **AVL tree** containing N elements (worst case)* O(N)

c) $f(N) = N (\log N)^2 + N \log \log N$ $O(N \log^2 N)$

d) $f(N) = N \log_2(N + N)$ $O(N \log N)$

e) *Enqueue in a **queue** containing N elements implemented using linked list nodes (worst case)* O(1)

f) *Determine what the maximum value is in a **binary search tree** containing N elements (worst case)* O(N)

g) *DecreaseKey(k, v) on a **binary min heap** containing N elements. Assume you have a reference to the key k. v is the amount that k should be decreased. (worst case)* $O(\log N)$

h) *Merging two **binary min heaps** containing N elements each. (worst case)* O(N)

i) $T(N) = T(N-1) + 5$ O(N)

2. **(6 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`. Your answer should be as "tight" and "simple" as possible.

Showing your work is not required

Runtime:

I.
```
void sunny (int n, int sum) {
    for (int k = n; k > 0; k = k/2) {
        for (int j = n; j > 0; j--)
            sum++;
    }
}
```

$O(n \log n)$

II.
```
int funny (int n) {
    if (n < 100)
        return n - 2;
    else {
        return funny (n - 2);
    }
}
```

$O(n)$

III.
```
int happy (int n, int sum) {
    for (int k = 0; k < n; k = k++) {
        for (int i = 0; i < k; i++)
            sum++;
        for (int j = n * n; j > 0; j--)
            sum++;

    }
    return sum;
}
```

$O(n^3)$

3. (10 pts) Big-O, Big Ω, Big Θ

(2 pts each) For parts (a) – (e) circle whether the statement is true or false. You do not need to show any work or give an explanation.

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

TRUE / FALSE b) $50\ N^2 + N^3 + 10\ N^4 = \Theta(N^3)$

TRUE / FALSE c) $5\ N + (\frac{1}{2})\ N = \Omega(N \log N)$

TRUE / FALSE d) $N^3 + N^3 = O(N^6)$

TRUE / FALSE e) $5\ N \log N = \Omega(N^2)$

4. (6 pts) Recurrence Relationships -

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

$T(1) = 7.$

and

$T(N) = T(N-2) + 4$ for odd 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).

$$T(N) = T(N-2) + 4$$
$$= (T(N-4) + 4) + 4$$
$$= T(N-4) + 8$$
$$= (T(N-6) + 4) + 8$$
$$= T(N-6) + 12$$
$$= T(N-2k) + k * 4$$

Want: $N - 2K = 1$

$N - 1 = 2k$

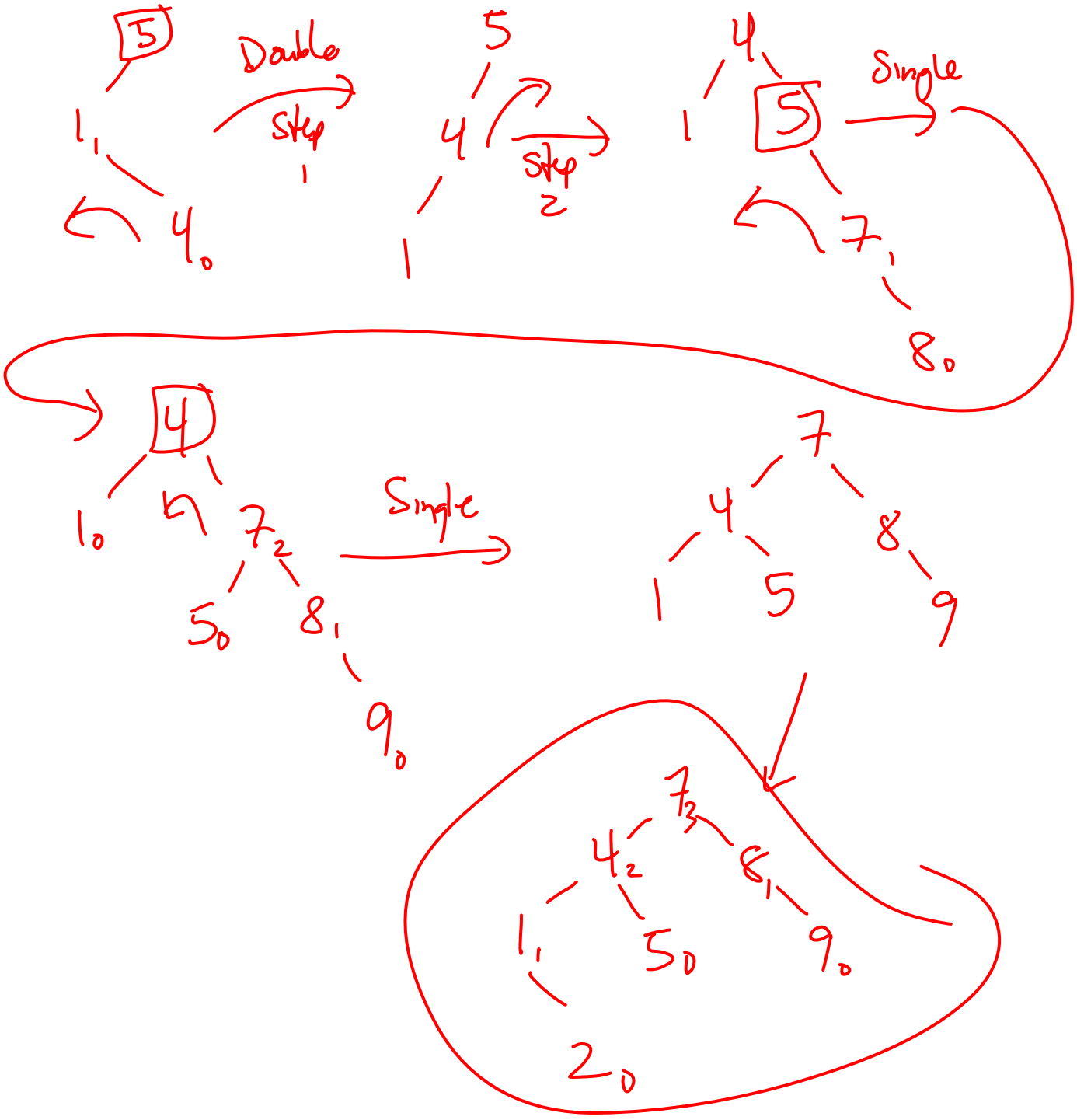
$K = \frac{N-1}{2}$

$$T(N) = T(1) + \left(\frac{N-1}{2}\right) * 4$$
$$= 7 + 2N - 2$$
$$T(N) = 2N + 5$$

Check:

$T(1) = 7$ —(+4)→ $T(3) = 2 \cdot 3 + 5 = 11$ —(+4)→ $T(5) = 2 \cdot 5 + 5 = 15$ —(+4)→ $T(7) = 2 \cdot 7 + 5 = 19$

5. (8 pts) Draw the AVL tree that results from inserting the keys 5, 1, 4, 7, 8, 9, 2 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.***



6. (6 pts) Trees

a) (2 pts) What is the **minimum 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*** not a formula.

$N = S(h-1) + S(h-2) + 1$

$= S(4) + S(3) + 1$

$= 12 + 7 + 1$

$= 20$

$S(0) = 1$

$S(1) = 2$

$S(2) = 4$

$S(3) = 7$

$S(4) = 12$

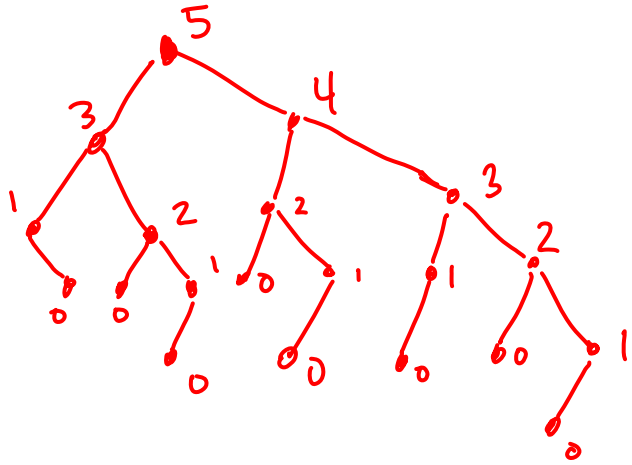
b) (2 pts) What is the **minimum number of nodes than must be examined in order to find the minimum value** in an **AVL tree of height 5?** ***Give an exact number*** not a formula.

3

c) (2 pts) Given a **perfect binary** tree of height h, what are the minimum and maximum **number of nodes in the right subtree** of the root?

Minimum: $2^h - 1$

Maximum: $2^h - 1$



7. (8 pts) B-trees

a) (4 pts) Given a B-tree of height H, give the tight big-O running time for a **find** operation in terms of M, L, N, and H. Do not simplify your answer – it is o.k. to ignore constants and smaller order terms other than those containing M, L, N, or H – but **do not remove any occurrences of M, L, N, or H from your answers**.

Worst case running time for a find operation =

$O(H \cdot \log_2 M + \log_2 L)$

internal nodes; binary search for correct ptr; binary search in leaf (LL also ok. for linear search)

Best case running time for a find operation =

$H \cdot 1 + 1 = O(H)$

b) (4 pts) Given the following parameters for a B-tree with M= 11 and L = 8:

k Key Size = 10 bytes

p Pointer Size = 2 bytes

Data Size = 16 bytes per record (***includes*** the key)

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

Internal node size $= M \cdot p + (M-1) \cdot k$

$= 11 \cdot 2 + 10 \cdot 10$

$= 22 + 100 = 122$

Leaf node size $= L \cdot$ data

$= 8 \cdot 16 = 2^7 = 128$

Max of these is 128 (+ also is power of 2)

so size of disk block is likely to be <u>128 bytes</u>.

8. (15 pts) Hash Tables

For a) and b) below, insert the following elements in this order: 51, 30, 39, 10, 19, 40. 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) Quadratic probing hash table

0	30
1	51
2	
3	19
4	10
5	
6	40
7	
8	
9	39

b) Separate chaining hash table – use an unsorted linked list for each bucket, insert at front

0	→ 40 → 10 → 30 /
1	→ 51 /
2	
3	
4	
5	
6	
7	
8	
9	→ 19 → 39 /

c) What is the load factor in Table a)? $\frac{6}{10} = \frac{3}{5}$

d) In a sentence or two, describe one advantage that **separate chaining** has over **quadratic probing**?

Many possible answers. Big difference is that sep. chaining will gradually degrade in performance and an insert will never "fail" – require re-hashing. while with quad. probing can fail on an insert & need to re-hash.

e) In a sentence or two, describe one advantage that **linear probing** has over **quadratic probing**?

If there are any open locations in the table you will eventually find one w. linear probing. With quad. probing empty spaces can exist and you could fail to find one.

9. (15 pts) Min Three Heaps - As discussed on homework 2, a three heap with *n* elements can be stored in an array A, where A[0] contains the root of the tree.

a) Given indexing that starts at A[0], give the equations for how you would calculate the three children of the value stored at index i as well as the parent.

Child 1: $(3*i)+1$ Child 2: $(3*i)+2$ Child 3: $(3*i)+3$

Parent: $\left\lfloor \frac{i-1}{3} \right\rfloor$

b) Implement the `insertThreeHeap` method. **To avoid ambiguity, please give your answer in Java.** You should implement a ***min*** heap. You can assume that the array `A` is big enough to contain all inserted values and that the values you will be inserting are `ints` > 0 and will not contain duplicates. You do not need to optimize your code by "bubbling a hole" up/down, but it is fine to do so. You may NOT assume that other methods have been implemented (such as percolateUp, percolateDown, swap, etc.), however it is fine to implement such helper methods if you would like to. Assume that you can read and modify the `size` of the heap.

```
// Given that values are stored in A[0] to A[size-1],
// insert val into the min three heap.
void insertThreeHeap(int[] A, int val) {
    size++;
    int i = percUp(A, size-1, val);
    A[i] = val;
}

int percUp(int[] A, int hole, int val) {
    while (hole > 0 && val < A[(hole-1)/3]) {
        A[hole] = A[(hole-1)/3];
        hole = (hole-1)/3;
    }
    return hole;
}
```

9. Min <u>Three</u> Heaps (continued) – feel free to use the space on this page as well.

10. (8 pts) B-tree Insertion

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

b) (2 pts) Based on the picture below, what are the values for M and L?

M = 4 L = 3

c) (4 pts) Starting with the B-tree shown below, insert **27**. Draw and circle the resulting tree (*including values for interior nodes*) below. Use the method for insertion described in lecture and used on homework.