Name:____

CSE 332, Spring 2010, Midterm Examination
30 April 2010

Please do not turn the page until the bell rings.

Rules:

- The exam is closed-book, closed-note. You may use a calculator *for basic arithmetic only.*
- **Please stop promptly at 3:20.**
- You can rip apart the pages, but please staple them back together before you leave.
- There are **100 points** total, distributed **unevenly** among **9** questions (many with multiple parts).

Advice:

- Read questions carefully. Understand a question before you start writing.
- **Write down thoughts and intermediate steps so you can get partial credit. But clearly circle your final answer.**
- The questions are not necessarily in order of difficulty. **Skip around.**
- If you have questions, ask.
- Relax. You are here to learn.

Name:____

1. (**8** points)

   For each function below, give an asymptotic upper bound using "big-Oh" notation. Your answer should be as "tight" and "simple" as possible.

   (a) $f(n) = 4n^3 + 5n^2 \log n$

   (b) $f(n) = 3n + 4 \log n + 2n$

   (c) $f(n) = 1.6^n + n^5$

   (d) $f(n) = 2 \log \log n + \log(n/2)$

   Solution:

   (a) $O(n^3)$

   (b) $O(n)$

   (c) $O(1.6^n)$

   (d) $O(\log n)$

Name:____

2. (**12** points) Consider this recurrence equation:

$$\begin{array}{rcll} T(1) & = & 7 & \\ T(n) & = & 4n + T(n-1) & \text{for } n > 1 \end{array}$$

(a) Prove that for all integers $n \geq 1$, $T(n) = 2n^2 + 2n + 3$

(b) True or false:

i. $T(n)$ is $O(n^2)$

ii. $T(n)$ is $\Omega(n^2)$

iii. $T(n)$ is $\Theta(n^2)$

Solution:

(a) Proof by induction on n

- Base case $n = 1$: $T(1) = 7 = 2 \cdot 1^2 + 2 \cdot 1 + 3$
- Inductive case $n = k + 1 > 1$:
  $T(n) = 4n + T(n-1)$ by definition
  $= 4n + 2(n-1)^2 + 2(n-1) + 3$ by induction
  $= 4n + 2n^2 - 4n + 2 + 2n - 2 + 3$ by factoring
  $= 2n^2 + 2n + 3$ by cancelling terms

(b) i. true

ii. true

iii. true

Name:____

3. (**16** points) Suppose there is a binary min-heap with exactly 4 nodes, containing items with priorities 5, 6, 7, and 8.

   (a) Show every possible binary min-heap that could match this description. For each, draw the appropriate tree *and* the array representation.

   (b) Argue briefly in English that your answer to part (a) is not missing anything, meaning no other binary min-heap is possible.

   (c) For *one* of your answers to part (a), show the result of 4 successive `deleteMin` operations. Clearly indicate which heap you are starting with. You can just draw the tree (not the array).

Solution:

(a) Three possibilities:

```
      5        -------------------          5        -------------------
     / \       |   | 5 | 6 | 7 | 8 |       / \       |   | 5 | 7 | 6 | 8 |
    6   7      -------------------        7   6      -------------------
   /                                     /
  8                                     8

      5        -------------------
     / \       |   | 5 | 6 | 8 | 7 |
    6   8      -------------------
   /
  7
```

(b) All 4-node heaps must have the shape of the 3 heaps in part (a). The minimum node, 5, must be at the root. The maximum node, 8, must be at a leaf. The only heap meeting these rules and not listed above does not satisfy the heap property because 7 is above 6:

```
      5
     / \
    7   8
   /
  6
```

(c)
```
      5                  6                   7
     / \               /   \                /
    6   7       --->  8     7       --->   8
   /
  8


--->    8       -->     <empty tree>
```

Name:____

4. (**10** points) Give a very short description of *what* the method `mystery` returns (not *how* it does it). Assume the `Node` is a binary search tree and make sure this is relevant to your answer.

```
class Node {
   int value;
   Node left;
   Node right;

   static int mystery(Node n) {
      if(n==null)
        throw new IllegalArgumentException();
      if(n.left == null)
        return leftmost(n.right);
      if(n.left.left == null && n.left.right == null)
        return n.value;
      return mystery(n.left);
   }

   // return leftmost descendant of n
   static int leftmost(Node n) {
      if(n==null)
        throw new IllegalArgumentException();
      if(n.left == null)
        return n.value;
      return leftmost(n.left);
   }
}
```

Solution:
`mystery` returns the second smallest value in the binary search tree rooted at `n` (throwing an exception if the tree has fewer than two elements).

Name:____

5. (**12** points)

   (a) Starting with an empty AVL tree, insert the following keys in order, clearly indicating the final tree: 5, 10, 8, 4, 6, 2

   (b) Beginning with the tree produced by part (a), show the AVL tree resulting from deleting 5, then 10, then 8.

Solution:

(a)

```
                           double
5  -->   5     -->  5        -->    8     -->     8      -->    8
          \           \            / \           / \           / \
          10           10         5  10         5   10        5   10
                      /                        /             / \
                     8                        4             4   6
```

```
                single
-->    8         -->           5
     /  \                    /   \
    5    10                 4     8
   / \                     /     /  \
  4   6                   2     6    10
 /
2
```

(b)

```
      5                     6                6                6             4
    /   \                 /  \              / \              /    single   / \
   4     8       -->     4    8     -->    4   8   -->      4      -->    2   6
  /     /  \            /      \          /                /
 2     6    10         2        10       2                2
```

```
A second simpler approach uses 5's predcessor at the first step
instead of its successor:
```

```
      5                     4                4             4
    /   \                 /  \              / \           / \
   4     8       -->     2    8     -->    2   8   -->   2   6
  /     /  \                 / \              /
 2     6    10              6   10           6
```

Name:____

6. (**12** points) Consider a B-Tree as defined in lecture and the textbook where $M = 64$ and $L = 6$ and the number of nodes is n. In English, describe an $O(\log_M n)$ algorithm for returning the 4^{th} smallest element of the tree. You may assume the root is not a leaf.

   Solution:
   Starting at the root, follow the left-most child of each node until the node's children are leaves. If the leftmost leaf has more than 3 elements, return element 3 of its array of data items. Else return element 0 of the array of data items in the leaf just to the right of the leftmost leaf.

Name:____

7. (**10** points) For both questions below, give an *exact* answer *and briefly justify your answer.*

   (a) What is the minimum number of data items in a B tree of height 4 (as defined in class and the textbook) with $M = 4$ and $L = 4$.

   (b) What is the minimum number of data items in an AVL tree of height 4?

   Solution:

   (a) We can use the equation from lecture or just rederive it for height 4 by putting the minimum number of children and data items everywhere: All internal nodes (including the root) can have as few as 2 children and the leaves can have as few as 2 data items. Because all leaves have to be at the same height, this is the same as a perfect binary tree of height 4 ($2^4 = 16$ leaves), with two data items at each leaf. So 32 data items.

   (b) Letting $S(h)$ be the minimum number of nodes in an AVL tree of height h, recall that $S(-1) = 0$, $S(0) = 1$ and for $h > 0$, $S(h) = 1 + S(h-1) + S(h-2)$. (This follows directly from the AVL balance property.) Therefore, we can simply calculate iteratively: $S(1) = 2$, $S(2) = 4$, $S(3) = 7$ and $S(4) = 12$. (You can also draw a minimal height-4 tree, but you need to justify that no height-4 tree with fewer nodes is balanced.) So 12 data items.

Name:____

8. (**13** points)

   Consider inserting data with integer keys 7, 9, 12, 18, 29 in that order into a hash table of size 11 where the hashing function is `h(key)%11`.

   - Show a chaining hash table after doing the insertions:

0	1	2	3	4	5	6	7	8	9	10

   - Show an open addressing with linear probing hash table after doing the insertions.

0	1	2	3	4	5	6	7	8	9	10

   - Show an open addressing with quadratic probing hash table after doing the insertions.

0	1	2	3	4	5	6	7	8	9	10

Solution:

(a) At index 1: a list holding 12
    At index 7: a list holding 29, then 18, then 7
    At index 9: a list holding 9

(b) `X 12 X X X X X 7 18 9 29`

(c) `29 12 X X X X X 7 18 9  X`

Name:____

9. (**7** points)

   Consider these four implementation approaches for hash tables:

   - Separate chaining
   - Open addressing with linear probing
   - Open addressing with quadratic probing
   - Open addressing with double hashing

   (a) Suppose many items for a hash table are initially hashing to the same table index. Which implementation approach or approaches are likely to lead to very poor performance?

   (b) Suppose many items for a hash table are initially hashing such that the first two hash to index i, the next two to index $i + 1$, the next two to index $i + 2$, etc. Which implementation approach or approaches are likely to lead to very poor performance?

   (c) Suppose you are trying to use various hash table libraries for some data you have, but you are getting poor performance for all of them. What should you do to try to get better hash table performance?

   Solution:

   (a) Separate chaining, linear probing, and quadratic probing

   (b) Linear probing

   (c) Change your hash function