CSE 373 Spring 2008: Midterm #1
(closed book, closed notes, NO calculators allowed)

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

Note: For questions where you are drawing pictures, please circle your final answer for any credit. There is one extra page at the end of the exam that you may use for extra space on any problem. If you detach this page it must still be turned in with your exam when you leave.

Good Luck!

Total: 77 points. Time: 50 minutes.

<table>
<thead>
<tr>
<th>Question</th>
<th>Max Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>77</strong></td>
<td></td>
</tr>
</tbody>
</table>
1. (14 pts) **Big-O**

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). 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) – g)):

\[
O(N^2), O(N^3 \log N), O(N \log N), O(N^2 \log N), O(N^5), O(2^N), O(N^3),
O(\log N), O(1), O(N^5), O(N^N)
\]

You do not need to explain your answer.

a) \( f(N) = N \log(N^2) + N \)  
\[ O(N \log N) \]

b) \( f(N) = 20000^2 + N \log N + N \)  
\[ O(N \log N) \]

c) \( f(N) = N^{3/2} + N^{3/2} \)  
\[ O(N^{1.5}) \]

d) \( f(N) = 100N + N \log N + N/2 \)  
\[ O(N \log N) \]

e) \( f(N) = N \cdot (\log (N^4) - \log N) + N^2 \)  
\[ O(N^2) \]

f) \( f(N) = N^2 \cdot \log_2 (N/2) + N^2 \)  
\[ O(N^2 \log N) \]

g) \( f(N) = \sqrt{2}(N^{1/3}) + \log_2 N \)  
\[ O(N) \]
2. (8 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 $n$. **Showing your work is not required** (although showing work may allow some partial credit in the case your answer is wrong – don’t spend a lot of time showing your work.). 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 I. – IV.):

- $O(n^2)$
- $O(n^3 \log n)$
- $O(n \log n)$
- $O(n)$
- $O(n^2 \log n)$
- $O(n^5)$
- $O(2^n)$
- $O(n^3)$
- $O(\log n)$
- $O(1)$
- $O(n^4)$
- $O(n^n)$

I. ```java
void silly(int n) {
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < i; ++j) {
            System.out.println("j = " + j);
        }
        for (int k = 0; k < n * 3; ++k) {
            System.out.println("k = " + k);
        }
    }
}
```

**Runtime:** $O(n^2)$

II. ```java
void silly(int n, int x, int y) {
    for (int i = 0; i < n; ++i) {
        if (x < y)
            for (int k = 0; k < n * n; ++k) {
                System.out.println("k = " + k);
            }
        else
            System.out.println("i = " + i);
    }
}
```

**Runtime:** $O(n^3)$

III. ```java
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n-1);
}
```

**Runtime:** $O(n)$

IV. ```java
void silly(int n) {
    if (n <= 0) return;
    System.out.println("n = " + n);
    silly(n/2);
}
```

**Runtime:** $O(\log n)$
3. (13 pts) **Read each question carefully.** No explanations are necessary, but explanations may yield partial credit. You must solve any summations or recurrences for full credit.

   a) (3 pts) What is the minimum and maximum number of nodes at depth \( d \) in a complete binary tree? (Hint: the root is at depth 0)

   Minimum = \( \frac{1}{2} \)

   Maximum = \( 2^d \)

   b) (3 pts) What is the minimum and maximum number of nodes in the right subtree of the root of a perfect binary tree, where the height of the entire tree (not the subtree) is \( h \) and \( h \geq 1 \)?

   Minimum = \( 2^{h-1} \)

   Maximum = \( 2^{h-1} \)

   c) (2 pts) Given an AVL tree containing 31 nodes, after inserting the 32\(^{nd}\) value, it is possible that we might have to do three or more rotations in order to maintain the balance property.

   (circle one) **TRUE** or **FALSE**

   d) (2 pts) A complete binary tree satisfies the AVL balance condition at each node.

   (circle one) **TRUE** or **FALSE**

   e) (3 pts) Given an unsorted array of size \( N \), each insert operation will write a value into the array at the next available location. The index of this location is stored in the integer variable `next_loc`. `next_loc` is updated (incremented) after each insert. Once the array is full, the \( N+1 \)\(^{st}\) insertion will cause a new array of size \( 2N \) to be created, and all \( N+1 \) values to be copied into the new array. What is the amortized cost of an individual insert operation? (circle one)

   \( O(N) \) or \( O(N^2) \) or \( O(1) \) or \( O(\log N) \) or \( O(N \log N) \)
4. (12 pts) **Trees**
   a.) (6 pts) Give traversals of the tree shown below:
      
      Pre-Order: 20, 11, 8, 9, 18, 42, 56
      Post-Order: 8, 9, 11, 18, 42, 56, 20
      In-Order: 8, 9, 11, 18, 20, 42, 56
   b.) (6 pts) Given the following tree:

   ![Tree Diagram]

   Circle **yes** or **no** to indicate whether the tree above might represent each of the following data structures. If you circle **no**, you must describe one way to modify the tree in order to make the answer into a yes.

   - AVL tree [**yes**] **no**
   - Binary Search Tree [**yes**] **no**
   - Full Binary Tree [**yes**] **no**

   **Node 8 and node 42 both need to have left children (all nodes must have zero or 2 children).**
5. (6 pts) **AVL Trees** Draw the AVL tree that results from inserting the keys: 2, 3, 5, 9, 6, 4 in that order into an initially empty AVL tree. You are only required to show your final tree. However if you draw intermediate trees, **please circle your final tree for ANY credit.**
6. (18 pts) **Running Time Analysis**: Give the tightest possible upper bound for the **worst case** running time for each of the following in terms of \( N \). 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) – f):

\[
O(N^2), O(N^3 \log N), O(N \log N), O(N), O(N^2 \log N), O(N^5), O(2^N), O(N^3), O(\log N), O(1), O(N^4), O(N^R)
\]

**For any credit, you must explain your answer.** Assume that the most time-efficient implementation is used. Assume that all keys are distinct.

a) Delete in a Binary Search Tree of size \( N \)

**Explanation:**

Worst case depth of a node = \( N \)

Time \( O(N) \) to find the node.

b) Finding the minimum value in a Splay Tree of size \( N \)

**Explanation:**

Worst case depth of a node = \( N \)

Time \( O(N) \) to find the leftmost node (spaying up takes time \( O(N) \) too)

(c) Inserting a value into an AVL tree of size \( N \), where the value you are inserting is smaller than any other values currently in the tree.

**Explanation:**

Worst case height of an AVL tree (depth of a node) = \( O(\log N) \). Time to find proper location = \( O(\log N) \) + \( \Delta \) rotation to fix.

d) Pop on a stack containing \( N \) elements implemented as a singly-linked list

**Explanation:**

Simply update a constant \( \pm \) of pointers:

```plaintext
temp = top, top = top -> next
return temp
```

e) Post-order traversal of a Binary Search Tree containing \( N \) elements

**Explanation:**

Must visit each node \( O(1) \) times

f) In-order traversal of an AVL Tree containing \( N \) elements.

**Explanation:**

Same as above. Order of traversal + balance does not matter.
7. (6 pts) Splay Trees

Draw the splay tree that results from inserting the value 13 into the splay tree shown below. Circle your final answer.