CSC 226 FALL 2016
ALGORITHMS AND DATA STRUCTURES II
ASSIGNMENT 1 - WRITTEN
UNIVERSITY OF VICTORIA

1. Consider the insertion of items with the following keys (in the given order) into an initially empty AVL tree: 30, 40, 24, 58, 48, 26, 11, 13, 17. Draw the resulting AVL trees after each insertion (nine of them).

2. What is the minimum number of internal nodes in an AVL tree of height 5? Draw such a tree.

3. Given functions $f, g: \mathbb{N} \to \mathbb{R}$, prove that $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

4. Consider the version of selection sort I gave you in lecture 2, shown below:

```
Algorithm selectionSort(A,n):
    Input: Array A of size n
    Output: Array A sorted
    for $k \leftarrow 0$ to $n-2$ do
    min $\leftarrow k$
            for $j \leftarrow k+1$ to $n-1$ do
                    if $A[j] < A[min]$ then
                            $min \leftarrow j$
                    end
            end
            swap($A[k]$, $A[min]$)
    end
end
```

   Draw the decision tree associated with running this algorithm on a sequence of 3 distinct values, say $S = [x_0, x_1, x_2]$, where each $x_i \in \{1,2,3\}$. Label each internal node with the comparison done at that node in terms of x_i (for example, $x_i < x_j$) and label each external node with the actual sequence S that leads the algorithm towards that external node. Note, your results should be consistent, meaning each path can correspond to at most one permutation of the original data.

5. Suppose you would like to sort n music files, but you only have an old, unreliable computer, which you have nicknamed "Rustbucket". Every time Rustbucket compares two music files, x and y, there is an independent 50-50 chance that it has an internal disk fault and returns the value -1, instead of the correct result of 1 for **true** or 0 for **false**, to the question, "$x \leq y$?" Otherwise, Rustbucket correctly performs every other kind of operation (including comparisons not involving music files.) Describe an efficient algorithm that can use Rustbucket to sort n music files correctly and show that your algorithm has *expected* running time that is $O(n \log n)$.