1. [15 points] Start with an empty Binary Search Tree. Draw a picture of the tree that will result from adding the following seven values to your tree, in the order they are given: 20 30 5 7 42 9 8
2. [5 points] What is the Balance Condition for AVL trees?

3. [15 points] Start with the AVL tree shown below. Draw the AVL tree that will result from adding, in order, the following values: 1 8 9 15
4. [15 points] Here is an array of 9 integer values:

\[
\begin{array}{cccccc}
12 & 9 & 4 & 1 & 7 & 6 & 8 & 3 & 2
\end{array}
\]

We can think of this as a tree. Give the array that will result from turning this array into a heap using the linear-time algorithm we developed in class. You may find it easier to think in terms of trees, but put your final answer back into an array.
5. [20 points] For a text analysis project I need a map where the keys are words and the value associated with a word is the number of times that word appears in a text sample. So <"bob", 286> is a typical <key, value> pair. I could implement this as a hashmap with separate chaining (linked lists) you did in Lab 8, or as a hashmap with linear open addressing as we did in class, or as a treemap with an AVL tree. I will put n <key, value> pairs into this structure. The hashmaps use an array of size M.

a) What are Big-Oh estimates of the worst-case and average case lookup times for a word that is in the map, for each of these structures.

<table>
<thead>
<tr>
<th></th>
<th>Average Case</th>
<th>Worst Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hashmap, chained</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hashmap, linear open address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Treemap</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Suppose I want to output the words in alphabetical order; which structure makes that easiest? Why?

c) Suppose I want to output the words in order from the most frequent to the least frequent. Which structure makes that easiest? How would you do that?
6. [15 points] As we discussed in class, Tries can use a lot of memory. One way around this is to use a trie for the first k levels (where k might be 3 or 4 or 5) and then have the leaf nodes store lists of all of the words with the same prefix. Here is such a structure with k == 2 that stores “bob”, “bobby”, “bobbin”, “blue”, “black”, “bluster” and “crazy”.

How would you create classes to make such a structure? Give class headers and class variables for such a trie. You don’t need to specify the methods, just the data variables. Don’t over-think this; I am just asking how to make a tree where the leaves have different data than the internal nodes.
7. [15 points] Give an algorithm (it is sufficient to explain it in English, but you can give pseudo-code if you wish) for the Binary Search Tree method \( \text{V findNext(K x)} \) which returns the value associated with the smallest key in the tree that is greater than \( x \). If there is no key larger than \( x \), return null. For example, with the following tree where the keys and values are the same

![Tree Diagram](image)

\( \text{findNext}(5) \) returns 10, \( \text{findNext}(18) \) returns 20, \( \text{findNext}(20) \) returns 30 and \( \text{findNext}(50) \) returns null. If the tree contains \( n \) nodes and is balanced (for example, if it is an AVL tree), estimate the running time for \( \text{findNext(key)} \).
[This page is for scratch work]
Please write and sign the Honor Pledge at the end of your solutions.