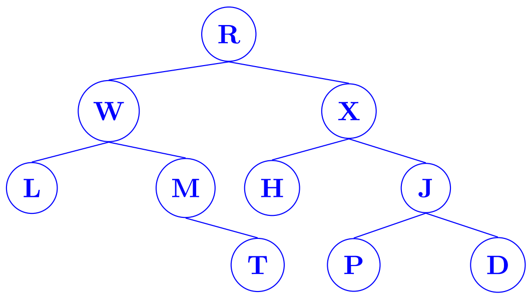
1. (2 points) I have a binary tree (**not** a binary search tree) with the following properties:

   - Each non-null node of T contains a single character
   - An in-order traversal of the tree reads "LWMTRHXPJD"
   - A post-order traversal of the tree reads "LTMWHPDJXR"
     This is a reminder that this is not a binary search tree.

   Give a pre-order traversal of the tree.

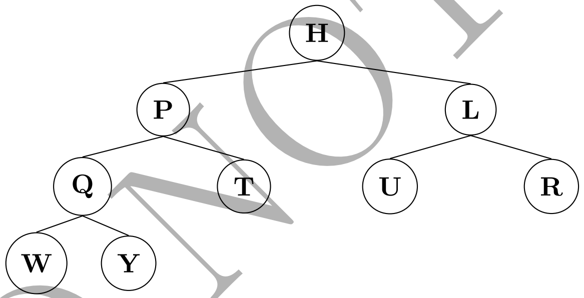
   A pre-order traversal of the tree reads: R W L M T X H J P D

   The post-order:

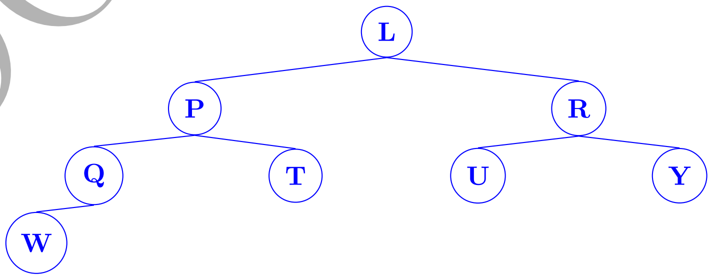


2. (1.5 point) In lecture, we saw that an array can be considered a complete tree and, if the heap property applies, it is also a heap.

   (a) Here is a binary heap, drawn as a tree. What is the character array/string representation of the heap? Write only the nine characters in the form; please do not write punctuation or arrows in the box.

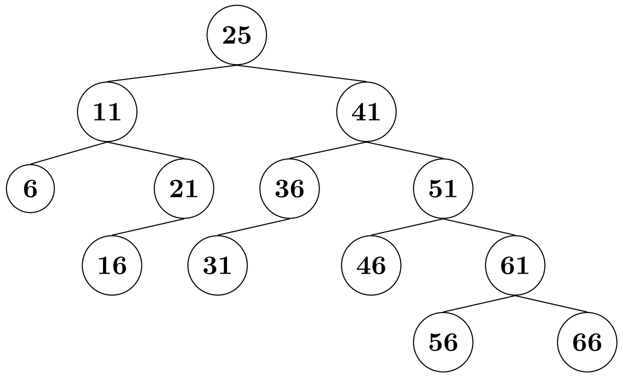


   (b) What is the result of a `remove-min` operation on this heap? You may want to draw the resulting heap, and on the answer page, you will describe the result.



   (c) What is the string/vector representation of the resulting heap?
   And if we do a remove-min, we get : LPRQTUYW

3. (1 point) Consider the following AVL Tree.



(a) Suppose I delete the key ‘46.’ How many re-balance operations will happen? 1 (51 becomes unbalanced)

(b) Suppose I delete the key ‘6,’ starting at the tree as pictured. How many re-balance operations will happen? 2. First at 11, but then at 25.

4. (2 points) Suppose you have a binary search tree (that may or may not be an AVL Tree) with the following struct declaration:

```
template<typename Key, typename Value>
struct TreeNode
{
    TreeNode<Key,Value> * left;
    TreeNode<Key,Value> * right;
    TreeNode<Key,Value> * parent;
    Key k;
    Value v;
};
```

Write a function that, given a pointer to a TreeNode, returns the Key of the in-order successor to that node. That is, return the smallest key in the tree that is larger than the provided one. If the parameter is the largest key in the tree, throw an exception.

For full credit, your response should have expected running time of $\mathcal{O}(\log n)$, although you are not required to analyze its time for this question. Answers that are guaranteed to take linear time will earn a zero.

`Key successor(TreeNode<Key,Value> * r) const`

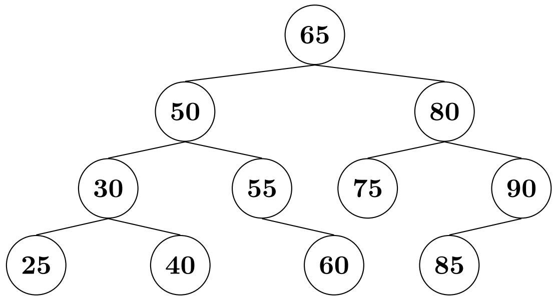
Here’s the general ideas behind an answer. Note that we don’t particularly care about syntax in these; for example, if you write (code or English) for the idea “number of nodes in the left-hand subtree,” that’s fine – even though in code you’d have to check for a `nullptr`.

If r has a right pointer that isn’t `nullptr`, return the smallest element in the right-hand subtree.

Otherwise, follow r’s parent pointers until you discover that one (including possibly r itself) is the left-child of its parent. The parent of that is the successor.

If you discover no such node and hit a `nullptr` from parent (which can only happen at the root), then r must have been the right-most key in the entire tree, which would be a maximum, so we throw an exception.

5. (1 point) Consider the following Binary Search Tree, being balanced via the rules of AVL-Trees. The keys are all an integer type. Identify a key we could insert into this tree that will cause the need for a rebalance operation. Write only a single integer in the box on the answer form.



There are many correct answers to this question. Anything in the range $[56, 64]$ except 60 will cause 55 to become unbalanced. Similarly, anything in the range $[81, 89]$ except 85 will cause 90 to become unbalanced.

6. (1 point) For purposes of this question, an AVL Tree with only a root is considered to be of height zero. The fewest nodes an AVL Tree can have, while having height zero, is one. The fewest nodes for height one is two. The fewest for height two is four, height three is seven, and height four is 12.

   We can continue and discover that the fewest nodes for height nine is 143 and height 10 is 232. What is the fewest for height 11? $376 = 1 + 143 + 232$ You may write your response either as an integer or an expression that evaluates to an integer. The fewest nodes an AVL Tree can have while having height h is $n_h = n_{h-1} + n_{h-2} + 1$. In class, we saw this with the proof of AVL height.

   It is possible your exam did not have this question; if that is the case and you wish to discuss it, please see the professor to do so.

7. (2 points) Suppose you have two min-heaps, A and B, with a total of n elements between them. You want to discover if A and B have a key in common. Give a solution to this problem that takes time $\mathcal{O}(n \log n)$. For this problem, do not use the fact that heaps are arrays. Rather, use the API of a heap (e.g., calls to `A.min()` or `B.extractMin()`). For full credit, your solution should use $\mathcal{O}(1)$ additional memory.

   Give a brief explanation for why your code has the required running time.

   Compare the min element of A and B. If they're equal, we're done. Otherwise, remove the smaller one (as it can't be in the other heap, as it's smaller than that heap's smallest element). If either heap is empty, stop and return false. Otherwise, repeat.

   This will iterate at most n times, and each iteration takes $\mathcal{O}(\log n)$ time, for a total of $\mathcal{O}(n \log n)$.