CSE 373 QuickCheck 2 Name: Evil Kevin Student ID: 1234567890

For answers that involve filling-in a ☐, fill-in the shape completely: ■.

1. Mark all of the following true expressions about the height of a binary search tree of size N.

☐ $O(\log N)$	☐ $\Theta(\log N)$	■ $\Omega(\log N)$
■ $O(N)$	☐ $\Theta(N)$	☐ $\Omega(N)$

Comprehending. In class, we discussed the problem with binary search trees: depending on the order we insert items into the tree, they are sometimes bushy but other times spindly. This should clue us into performing case analysis. Because the question does not specify the shape of the tree nor specify worst case or best case analysis, we need to do separate case analysis and then summarize the results for the overall order of growth for the height of the tree.

Modeling: Best Case. The best case is the smallest order of growth: in this case, the smallest height. This occurs when the tree is bushy. This is logarithmic because, given a bushy tree of size N, there are about $N/2$ items on the last level, $N/4$ items on the level above the last level, $N/8$ items on the level above that… This continues until the root where we have 1 item.

How many levels are in this tree? We know there are N total items, divided into levels with the following expression.

$$\frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \cdots + 4 + 2 + 1$$

How many levels (different terms) are in this equation? There are almost exactly $\log_2 N$ of these terms.

Modeling: Worst Case. The worst case is the greatest order of growth: in this case, the greatest height. This occurs when the tree is spindly. This height grows exactly linear with the number of items in the tree since it's just like an `OrderedLinkedSet`, hence the order of growth of the height in the worst case is $\Theta(N)$.

Formalizing: Overall. Since this question did not specify best or worst case, we should give our answer with respect to all possible scenarios, best and worst case inclusive. The height of any binary search tree of size N can be anywhere between $\log_2 N$ and N. The Change of Base formula tells us that $\log_2 N \approx \log N$ since all logarithm bases are a constant factor from each other.

Therefore, the overall order of growth is bounded below by $\Omega(\log N)$ and above by $O(N)$.

2. Mark all of the following true expressions about the height of a B-tree of size N.

■ $O(\log N)$	■ $\Theta(\log N)$	■ $\Omega(\log N)$
■ $O(N)$	☐ $\Theta(N)$	☐ $\Omega(N)$

This problem is somewhat simpler than the first question since B-trees are always balanced!

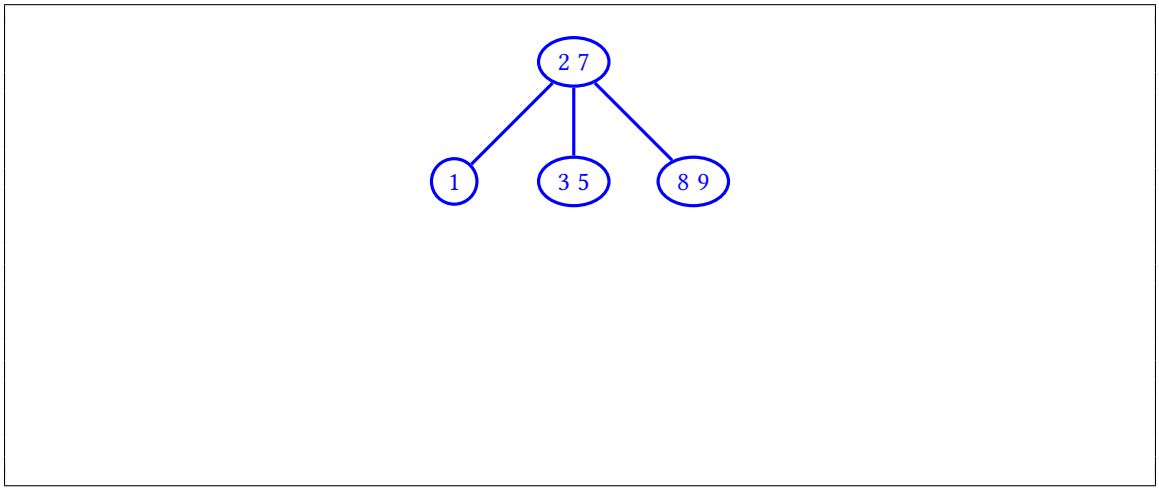
Comprehending. In class, we discussed that B-trees are always balanced.

Modeling. More specifically, their height is always about somewhere between $\log_2 N$ (all 2-nodes), $\log_3 N$ (all 3-nodes), and $\log_4 N$ (all 4-nodes). All possible 2-3-4 tree structures are therefore between height $\log_2 N$ and $\log_4 N$.

Technically, there is a case analysis here. If our tree has all 2-nodes, then the height is about $\log_2 N$, which is the worst case (greatest possible) height. If our tree has all 4-nodes, then the height is about $\log_4 N$, which is the best case (smallest possible) height.

Formalizing: Overall. The Change of Base formula tells us that $\log_2 N \approx \log_3 N \approx \log_4 N \approx \log N$ since all logarithm bases are a constant factor from each other. The order of growth of the height of a B-tree of size N is in $\Theta(\log N)$. This also implies that the order of growth of the height is also in $O(\log N)$ and $\Omega(\log N)$. But the height is also in $O(N)$ since all logarithms are bounded above by linear functions.

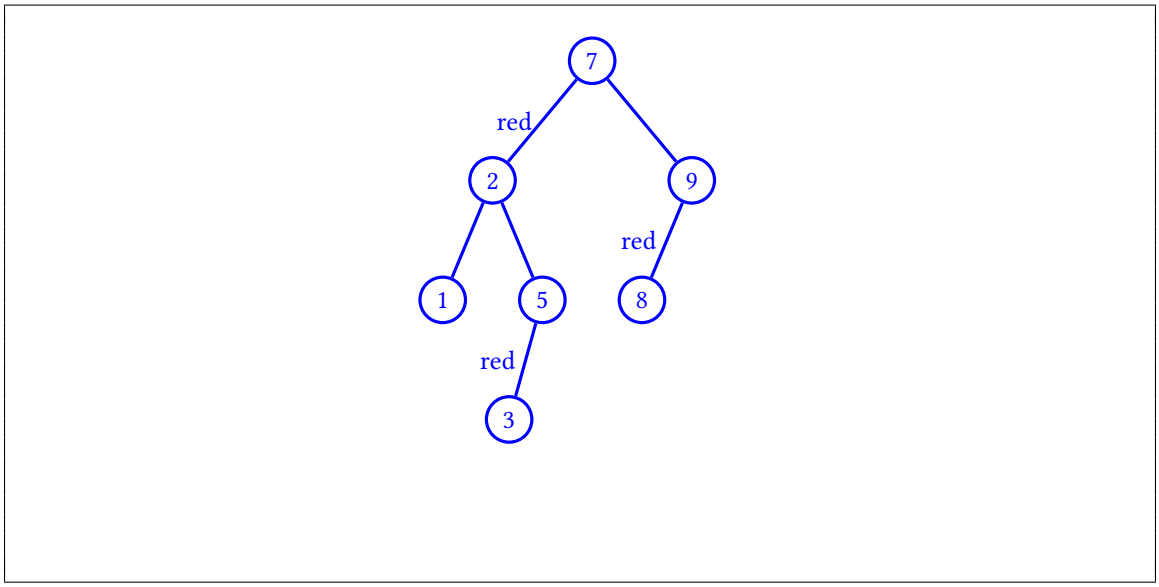
3. Draw the 2-3 tree that results from inserting the following items in this order: 1, 2, 3, 7, 8, 9, 5.



Use the B-Tree Visualization with Max-Degree = 3.

`https://www.cs.usfca.edu/~galles/visualization/BTree.html`

4. Draw the corresponding left-leaning red-black tree. Write "red" next to red links.



Left-leaning red-black trees (LLRB trees) have a 1-1 correspondence with B-trees. Use the B-Tree Visualization with Max. Degree = 3 first before attempting to understand this problem.

`https://www.cs.usfca.edu/~galles/visualization/BTree.html`

2-3 trees have 2 kinds of nodes: 2-nodes and 3-nodes. 2-nodes translate easily: we don't need to do anything about them. 3-nodes need to split up into 2x 2-nodes with a left-leaning red-link connecting them.

Many mistakes are due to having the wrong 2-3 tree! In addition, in an LLRB tree, we can never have a right-leaning red link!