1. (a) [5 marks] Prove or disprove: \(2^{\lg n} + n \in \Theta(n)\).
   (b) [5 marks] Prove or disprove: \(2^{2\lg n} \in \Theta(n)\).

2. This question is about proving a useful theorem from the textbook:
   If \(f(n) \in O(g(n))\) and \(g(n) \in O(h(n))\), then \(f(n) \in O(h(n))\).
   Assume \(f(n) \geq 0\), \(g(n) \geq 0\), and \(h(n) \geq 0\) for all natural \(n\).

   (a) [2 marks] My friend attempts the proof below. What goes wrong?
   From the assumptions, \(\lim_{n \to \infty} \frac{f(n)}{g(n)}\) and \(\lim_{n \to \infty} \frac{g(n)}{h(n)}\) exist and are finite.
   Hence
   \[
   \left(\lim_{n \to \infty} \frac{f(n)}{g(n)}\right) \left(\lim_{n \to \infty} \frac{g(n)}{h(n)}\right) = \lim_{n \to \infty} \frac{f(n) g(n)}{g(n) h(n)} = \lim_{n \to \infty} \frac{f(n)}{h(n)}
   \]
   exists and is finite.
   Therefore \(f(n) \in O(h(n))\).

   (b) [8 marks] Prove the theorem.

3. [5 marks] Insert 16 into the following weight-balanced binary search tree. Rebalance. Hand in: (i) the tree after inserting but before rebalancing; (ii) the name of the rotation necessary—“single” or “double”; (iii) the tree after rebalancing.

4. [5 marks] Delete 75 from the following weight-balanced binary search tree. Rebalance. Hand in: (i) the tree after deleting but before rebalancing; (ii) the name of the rotation necessary—“single” or “double”; (iii) the tree after rebalancing.
5. [14 marks] Prove that in Introduction to Weight-balanced BSTs, in subcase 1, the single-rotation restores balance.

More precisely, assume these before the rotation (the picture on the left):

(a) \( v \) is unbalanced because \( v \)'s right subtree is too large.

(b) If \( v \)'s right subtree had one fewer node, \( v \) would be balanced. (I.e., the unbalance is caused by adding one node recently.)

(c) We are in subcase 1, i.e., \( \text{size}(x.left) + 1 < \text{size}(x.right) + 1 \times 2 \).

(d) \( x \) is balanced.

Then prove these after the rotation (the picture on the right): the new \( v \) is balanced, and the new \( x \) is balanced.

Suggestion: carefully write down the inequalities corresponding to the assumptions, label them, then refer to them in the rest of the proof.

Note on marking: Coherence is as important as validity: A disorganized bunch of true statements is as wrong as a wrong proof.

6. [12 marks] Implement insertion and rebalancing for weight-balanced binary search trees. The starter code and the test program are on Blackboard: WBT.java, Node.java, and TestWBT.java. You will add your code to only WBT.java and submit it.

Package declaration All files declare “package WBT;”. Please try not to change it. We are about 60 students altogether, and it would be impractical to cater for 60 variations.

I declare a package name because I suspect that some of you use IDEs that force you to declare a package name. So let’s all stick to the same name.
If you use the command line instead of an IDE (I confess I’m primitive and I still do this), the package declaration means that you need to: put the source code under subdirectory WBT, call the compiler inside, but run the test program outside.

**How you can test**  If compilation is successful, you can use “java WBT.TestWBT” or IDE equivalents to run my test program. This runs all tests sequentially, but aborts at the first failure. Failures are represented by exceptions with error messages.

You can use “java WBT.TestWBT 1” to run just test #1, for example. See also the “main” method and its comment in the file.

**Marking scheme**  If your code looks like an attempt at the required algorithm (as opposed to an attempt at customizing for my test cases), you get 4 marks to start. In addition:

- If your code compiles: 1 more mark per test passed. But watch these requirements:
  - I will compile with these strict options, which mean that any unsafe type cast (which you don’t need for this task) is considered a hard error:
  
  ```
  javac -Xlint:unchecked -Werror WBT.java
  ```
  I will give each test case 1 second only on the Mathlab server. Timing out is a failure. Each test case is timed separately. My tests are like 10 nodes each, so taking more than 1 second is a clear sign of some circular problem.
- If your code does not compile in the sense above: 0 more marks.

If your code does not look like an attempt at the required algorithm, 0 to 3 marks depending on how far o

**How and what to submit**  Please both print out WBT.java (only) on paper and submit online.

To submit online:

(a) Upload to your account on the server mathlab.utsc.utoronto.ca. From linux: scp or sftp. From IITS Windows workstations: WinSCP. (You may like to put WinSCP on your own Windows computer too.)

(b) Login to mathlab.utsc.utoronto.ca. From linux: ssh. From IITS Windows: PuTTY. (You may like to put PuTTY on your own Windows computer too.)

(c) `submit -c cscb63s15 -a A1 WBT.java`

(d) You can verify by `submit -l -c cscb63s15 -a A1`

You can re-submit by `submit -f -c cscb63s15 -a A1 WBT.java`

Don’t forget to also print it out on paper and hand in with your other paper answers.