1. Draw the binary tree whose preorder traversal is $bgaeidcfh$ and whose inorder traversal is $agiebdcfh$. [5 points]

2. The **balance factor** of an internal node $v$ of a binary tree is the difference between the heights of the left and right subtrees of $v$. Write a recursive routine which will print the balance factors of all nodes in a binary tree. What is the running time of this routine? [6 points]

3. Consider an ordered tree $T$ and a binary tree $T'$ representing it, using the first-child next-sibling representation (section 10.4). An inorder traversal of $T'$ is equivalent to what kind of traversal of $T$? [4 points]

4. In class we defined the internal path length $I$ and the external path length $E$, both measures of a binary tree. If that tree has $n$ (internal) nodes, show that $E = I + 2n$. (This is exercise B.5-5, p 1180.) [8 points]

5. Consider the tree of Figure 12.2 on p 290. How many different permutations of the values it contains, when inserted in that order, will yield this particular tree? [8 points]

6. How many permutations of $1, 2, \ldots, n$ yield a skew tree? (Since any one skew tree is generated by just one permutation, this question is asking for the number of skew trees of $n$ nodes.) [5 points]

7. *(Search path splitting a BST)* Exercise 12.2-4, p 293. [4 points]

Total: 40 points

Notes:

- *(Q2)* Consider the following three formulas:
  
  - $height(null) = -1$
  
  - $height(p) = 1 + \max\{height(p.left), height(p.right)\}$
  
  - $balFac(p) = height(p.left) - height(p.right)$

  These suggest that you may want to compute the height and the balance factor at the same time. You may simply print out the balance factors, in any order.

- *(Q3)* To get $T'$, imagine the first-child as a left pointer and the next-sibling as a right pointer.
• (Q4) We had \( I = \sum_{v \in V} d(v) \), where \( V \) is the set of nodes and \( d(v) \) is the depth of a node. \( E \) is defined similarly, over all external nodes. You will want to use induction.

• (Q5) Consider a tree where
  
  – the left subtree contains \( n \) nodes and is generated by \( r \) permutations
  – the right subtree contains \( m \) nodes and is generated by \( s \) permutations

Then the whole tree contains \( n + m + 1 \) nodes and is generated by \( r \cdot s \cdot \binom{n+m}{n} \) permutations.