5.1*a* A general tree does not differentiate between children, while a binary tree with a single left sub-tree is different from one where that sub-tree is the right sub-tree.

5.1*b* An O(h) function will have to perform possibly one more function call, while an O(n) operation will have to perform O(n) additional function calls.

5.1*c* The least number is 1, while the greatest number is $\left\lceil \frac{n}{2} \right\rceil$.

5.1*e* A full binary tree always has an even number of nodes, and the number of leaf nodes is always (n + 1)/2.

5.1*g* Write a member function that returns the number of leaf nodes that are descendant from the node the member function is called on.

```
template <typename Type>
int Binary_node<Type>::leaf_count() const {
    if ( empty() ) {
        return 0;
    } else if ( is_leaf() ) {
        return 1;
    } else {
        return left()->leaf_count() + right()->leaf_count();
    }
}
```

5.1*h* The best-case scenario is that the change in bit codes for another letter with the same number of bits. The only other case is the worst-case: everything following that change will be completely garbled.

5.1*i* lg(4000) = lg(4) + lg(1000) ≈ 2 + 10 = 12,
lg(256000) = lg(1000000/4) = lg(1000000) – lg(4) ≈ 20 – 2 = 18
lg(8000000) = lg(8) + lg(1000000) ≈ 3 + 20 = 23