Question 1 (10 points):
How many nodes can there be in:

a) A complete binary tree of height h?

b) A perfect binary tree of height h?

c) A perfect ternary tree of height h? (In a ternary tree, each node may have up to three children.)

Question 2 (10 points):
For each of the following questions, suppose you have the following binary search tree.

```
      A
     / \  
    B   F
   / \  /  
  C   E D   G
```

a) Give the inorder traversal of the tree.

b) Draw the result of performing a single right rotation on the root of the tree.
Question 3 (15 points):
One way to represent a mathematical expression in a computer program is using an expression tree. Let's assume we have an expression containing only operators and numbers, such as:

\[ 3 + (1 - 4) * 5 / 6 \]

We can add parentheses to make the order of operations explicit:

\[ (3 + (((1 - 4) * 5) / 6)) \]

Now, we can store this calculation in a binary tree. Each number is a leaf node in the tree, and each operator is an interior node with two children (one for each of the operands, which could be numbers or more complicated subexpressions). Suppose you are given a node of the form:

class CalcNode {
    private float number;
    private char operator; // '+', '-', '*', '/'
    private CalcNode left, right;
    public boolean isOperator() { ... }

    public float performOperator(float operand1, float operand2) { .... }
    // Assume that you have getters and setters where necessary.
}

Write a method which evaluates the expression represented by that node.

float evaluate(CalcNode node) {

}

Question 4 (5 points):
The Day-Stout-Warren tree balancing algorithm does not affect the inorder traversal of the tree, despite completely rearranging its structure. Briefly explain why this is the case.
**Question 5 (15 points):**
You are given the following fragment of code.

```c
int mystery(int x, int n) {
    int result = 0;
    if (n == 0)
        result += 1;
    else {
        for (int i = 1; i <= x; i++)
            result += mystery(x, n - 1);
    }
    return result;
}
```

a) What is the big-Oh running time of this method?

b) Give a recurrence \( f(n) \) which describes the number of additions performed.

c) Find the closed form of your recurrence from part b.

**Question 6 (10 points):**

a) What property do nodes in a minimum heap have?

b) Explain why the largest value in a minimum heap *must* be in a leaf node.
**Question 7 (10 points):**
A Fibonacci tree is a tree which can be described visually as follows:

\[ F(0) \quad F(1) \quad F(n) \]
\[ \quad \circ \quad \circ \quad F(n-1) \quad F(n-2) \]

a) Write a recurrence which gives the number of nodes in a Fibonacci tree of height \( h \).
(You do not need to find the closed form.)

b) Explain how Fibonacci trees are related to AVL trees.

**Question 8 (15 points):**
When a node with two children is deleted from a binary search tree, it needs to be replaced with an appropriate node from somewhere else in the tree. Suppose you have the following node class:

```java
class Node {
    public String value;
    public Node left;
    public Node right;
}
```

a) Give the two places where a valid replacement for some node \( P \) could be found, and explain why these replacements work.
b) Write a method which locates a replacement given a reference to the deleted node.

```java
Node getReplacement(Node deletedNode) {
}
```

Question 9 (20 points):
A set is a collection in which no value may occur more than once, and the values occur in no particular order. It has the following basic interface:

```java
interface Set<E> {
    void add(E element);
    void remove(E element);
    boolean contains(E element);
}
```

a) Explain why implementing this interface with a hash table would be effective.

b) Tom notes that it would be useful to iterate through all of the values in a Set, and suggests making the data structure Iterable. The iterator may return the elements of the set in any order. Assuming our set is implemented with a hash table, what is the running time to perform such an iteration?

c) Dick proposes speeding up iteration by keeping a doubly-linked list alongside the hash table. Explain how he plans to use the list and describe the pros and cons of this modification.
d) Harriet decides that being able to iterate through the set elements in ascending order would be useful after all, and creates an OrderedSet interface. What should she use to implement it, and why?