Name: ______________________________

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1. (*10 Points*) State the precise mathematical definitions for $O(g(n))$, $\Omega(g(n))$, and $\Theta(g(n))$.

2. (*10 Points*) Prove that $f(n) = 2^{\sqrt{\lg(n)}}$ is $O(n^a)$ for any positive number a.

3. (*10 Points*) Find an exact closed form formula for the number of times the inner loop body is executed and state the computational complexity of the loop.

   (a)
   ```
   for (int cnt = 0, i = 1; i <= n; i *= 2)
        for (j = 1; j <= n; j++)
             cnt++;
   ```

   (b)
   ```
   for (int cnt = 0, i = 1; i <= n; i *= 2)
        for (j = 1; j <= i; j++)
             cnt++;
   ```

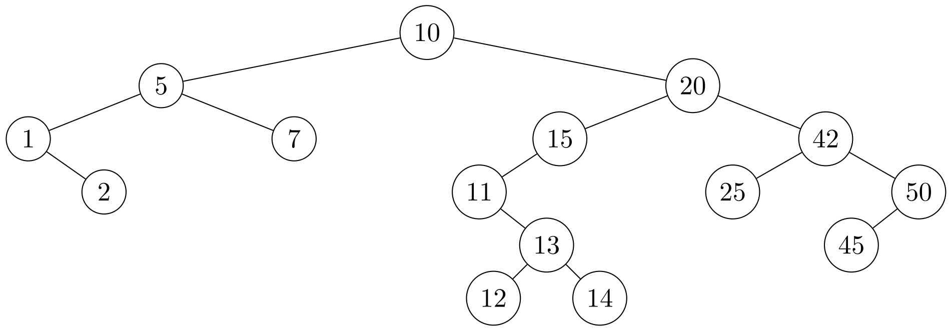
$$\sum_{i=1}^{n} 1 = n \qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \qquad \sum_{i=0}^{n-1} ar^i = \frac{a(1-r^n)}{1-r} = \frac{a(r^n-1)}{r-1}$$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \qquad \sum_{i=1}^{n} i^3 = \left(\frac{n(n+1)}{2}\right)^2$$

4. (*10 Points*) Draw the following tree after

   (a) A delete by merging of 10. Use the successor node.

   (b) A delete by copy of 10. Use the predecessor node.



For the remainder of the exercises you will be either implementing or adding functions to the standard templated binary search tree. The specification for the tree is below. You may assume that you have these functions at your disposal to use, unless you are asked to write the implementation to that function.

```
template<class T> class BinaryTree {
private:
    class TreeNode {
    public:
        T value;
        TreeNode *left;
        TreeNode *right;
        TreeNode() { left = right = nullptr; }
    };

    TreeNode *root;

    void insert(TreeNode*&, TreeNode*&);
    void destroySubTree(TreeNode*);
    void deleteNode(T, TreeNode*&);
    void makeDeletion(TreeNode*&);
    void displayInOrder(TreeNode*) const;
    void displayPreOrder(TreeNode*) const;
    void displayPostOrder(TreeNode*) const;

public:
    BinaryTree() { root = nullptr; }
    ~BinaryTree() { destroySubTree(root); }

    void insertNode(T);
    bool searchNode(T);
    void remove(T);

    void displayInOrder() const {
        displayInOrder(root);
    }

    void displayPreOrder() const {
        displayPreOrder(root);
    }

    void displayPostOrder() const {
        displayPostOrder(root);
    }
};
```

5. (*10 Points*) Implement the displayPreOrder recursive function for this class.

6. (*10 Points*) Write both the specification and implementation of an iterative preorder display function.

7. (*10 Points*) Write both the specification and implementation of a recursive function to count the number of leaves in a binary tree. Include a non-recursive function that initiates the recursive version that could be called from the main.

8. (*10 Points*) Write both the specification and implementation of an iterative function to count the number of leaves in a binary tree.

9. (*10 Points*) Write both the specification and implementation of a function that will load the values of a tree into a vector so that the resulting vector is sorted. If you write this recursively include a non-recursive function that initiates the recursive version that could be called from the main.

10. (*10 Points*) Write both the specification and implementation of a function that will create a complete binary tree with each node holding the level it is at, root level will be 1, it's children 2, and so on. The function should take in a single parameter that specifies the height of the final complete tree. You may assume that the tree is empty before calling this function. If you write this recursively include a non-recursive function that initiates the recursive version that could be called from the main.