Problem 1. Complete the following implementation of a function `above` that returns whether the largest element of a BST is larger than `x`.

```cpp
bool above(Node* root, int x) {
    if (_____ == nullptr)
        return false;
    if (root->x _____ x)
        return true;
    return above(root->_____, _____);
}
```

For a height-`h` BST with `n` nodes, the worst-case running time of `above` is \( \Theta(\text{function of } n \text{ and } h) \).

Problem 2. Complete the following implementation of a function that returns the number of elements in a BST (containing strings) that come lexicographically after `s`.

```cpp
int after(Node* root, string s) {
    if (root->x < s)
        return after(root->_____ , s);
    int lc = after(root->left, _____ );
    int rc = after(root->right, _____ );
    if (root->x _____ s)
        return lc + rc + _____;
    return lc + rc;
}
```

For a height-`h` BST with `n` nodes, the worst-case running time of `after` is \( \Theta(\text{function of } n \text{ and } h) \).
Problem 3. Determine the truth of the following statements about binary search trees with $n$ nodes.

- The node containing the median value is always the root. □ True □ False
- The node containing the largest value is never a leaf. □ True □ False
- If the root has no right child, then it contains the largest value. □ True □ False
- The maximum height of such a tree is $n - 1$. □ True □ False

Problem 4. Complete the following implementation of a function `wbalanced` (and helper function `weight`) that returns whether a binary tree is weight-balanced.

```cpp
int weight(Node* root) {
    if (root == nullptr)
        return 0;
    return weight(root->_____) + weight(root->_____) + 1;
}

bool wbalanced(Node* root) {
    if (root == nullptr)
        return true;
    int imbalance = weight(root->left) - weight(root->right);
    if (imbalance < -1 _____ imbalance > 1)
        return _____;
    return wbalanced(root->left) _____ wbalanced(root->right);
}
```

For a binary tree with $n$ nodes, the worst-case running time of `wbalanced` is $\Theta(\text{function of } n)$. 
