Problem 1. Complete the following function that prints the elements of the Nodes visited during a search for s. The elements should be printed in reverse order of their visit (i.e., the root should be printed last).

```cpp
void visited(Node* root, string s)
{
    if (root == nullptr)
        return;

    if (root->x == s)
    {
        cout << root->x << endl;
        return;
    }
    if (root->x < s)
        visited(root->_____ , s);
    else
        visited(root->_____ , s);
    cout << _____ << endl;
}
```

Problem 2. Fill in the blanks with answers based on AVL trees.

\[ f(h) = \text{function of } f(h-1), f(h-2), \ldots \text{ with } f(0) = \text{number}, f(1) = \text{number} \] is the recurrence relation describing the minimum number of nodes in an AVL tree of height \( h \).

The minimum number of nodes in an AVL tree of height 3 is \text{number}.

The maximum number of nodes in an AVL tree of height 3 is \text{number}.

The minimum number of nodes in an AVL tree of height 4 is \text{number}. 
**Problem 3.** Complete the following implementation of a function that performs a left rotation on the AVL tree rooted at `root`.

```cpp
void left_rotate(Node* root)
{
    Node* rc = root->right;
    root->_____ = rc->right;
    rc->right = _____;
    rc->left = _____;
    root->left = _____;
    swap(root->x, rc->x);
    update_lrot_heights(root);
}
```

**Problem 4.** Complete the following implementation of the function `update_lrot_heights` from Problem 3 (and helper function `update_node_height` that updates the heights of nodes after performing a left rotation).

```cpp
void update_node_height(Node* cur)
{
    if (cur->left != nullptr && cur->left->height > cur->height)
        cur->height = _____;
    if (cur->right != nullptr _____ cur->right->height > _____)
        cur->height = _____;
}

void update_lrot_heights(Node* root)
{
    update_node_height(root->_____);
    update_node_height(root);
}
```