```
    avl->nodes[pivot].left = x_index;
    avl->nodes[x_index].right = T2;

    // Update heights:
    avl->nodes[x_index].height = 1 + max(_avl_get_height(avl, avl->nodes[x_index].left),
_avl_get_height(avl, avl->nodes[x_index].right));
    avl->nodes[pivot].height = 1 + max(_avl_get_height(avl, avl->nodes[pivot].left), _avl_get_height(avl,
avl->nodes[pivot].right));
    return pivot;
}

static idx_t
_avl_insert_recursive(AVLTree *avl, idx_t node_index, int key) {

    if (node_index == IDX_INVALID) {
        idx_t new_index = avl->elements;
        AVLNode* new_node = &avl->nodes[new_index];
        new_node->key = key;
        new_node->left = IDX_INVALID;
        new_node->right = IDX_INVALID;
        new_node->height = 1;
        avl->elements++;
        return new_index;
    }

    /* Binary Search Tree */
    if (key < avl->nodes[node_index].key) {
        avl->nodes[node_index].left = _avl_insert_recursive(avl, avl->nodes[node_index].left, key);
    } else if (key > avl->nodes[node_index].key) {
        avl->nodes[node_index].right = _avl_insert_recursive(avl, avl->nodes[node_index].right, key);
    } else {
        return node_index;
    }

    avl->nodes[node_index].height = 1 + max(_avl_get_height(avl, avl->nodes[node_index].left),
                                            _avl_get_height(avl, avl->nodes[node_index].right));

    // Get balance factor to check if rebalancing is needed.
    int balance = _avl_get_balance(avl, node_index);

    /* Left Left */
    if (balance > 1 && key < avl->nodes[avl->nodes[node_index].left].key)
        return _avl_rotate_right(avl, node_index);

    /* Right Right */
    if (balance < -1 && key > avl->nodes[avl->nodes[node_index].right].key)
        return _avl_rotate_left(avl, node_index);

    /* Left Right */
    if (balance > 1 && key > avl->nodes[avl->nodes[node_index].left].key) {
        avl->nodes[node_index].left = _avl_rotate_left(avl, avl->nodes[node_index].left);
        return _avl_rotate_right(avl, node_index);
    }

    /* Right Left */
    if (balance < -1 && key < avl->nodes[avl->nodes[node_index].right].key) {
        avl->nodes[node_index].right = _avl_rotate_right(avl, avl->nodes[node_index].right);
        return _avl_rotate_left(avl, node_index);
    }

    return node_index;
```