```
    free(avl->nodes);
}

void
tree_avl_resize(AVLTree *avl) {
    idx_t new_capacity = avl->capacity*RESIZE_FACTOR;
    if (new_capacity >= IDX_INVALID) {
        perror("AVL tree exceeded maximum capacity.");
        exit(EXIT_FAILURE);
    }

    AVLNode* new_nodes = (AVLNode*) realloc(avl->nodes, sizeof(AVLNode)*new_capacity);
    if (new_nodes == NULL) {
        perror("Failed to allocate enough memory for tree resize.");
        exit(EXIT_FAILURE);
    }

    avl->nodes = new_nodes;
    avl->capacity = new_capacity;
}


static int
_avl_get_height(AVLTree* avl, idx_t index) {
    return (index == IDX_INVALID) ? 0 : avl->nodes[index].height;
}

static int
_avl_get_balance(AVLTree* avl, idx_t index) {
    if (index == IDX_INVALID) return 0;
    return _avl_get_height(avl, avl->nodes[index].left) - _avl_get_height(avl, avl->nodes[index].right);
}


static idx_t
_avl_rotate_right(AVLTree *avl, idx_t node_idx) {
    g_rotation_count++;

    idx_t pivot = avl->nodes[node_idx].left;
    idx_t T2 = avl->nodes[pivot].right;

    // Perform rotation:
    avl->nodes[pivot].right = node_idx;
    avl->nodes[node_idx].left = T2;

    // Update heights:
    avl->nodes[node_idx].height = 1 + max(_avl_get_height(avl, avl->nodes[node_idx].left),
_avl_get_height(avl, avl->nodes[node_idx].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_rotate_left(AVLTree *avl, idx_t x_index) {
    g_rotation_count++;

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

    // Perform rotation:
```