# test.cpp Code

| #include <catch2/catch\_test\_macros.hpp> #include "avl\_tree.h" #include <iostream> #include <vector> #include <string> #include <algorithm>   using namespace std;  // Blake McGahee // UFID: 82924917  // Test at least five incorrect commands. TEST\_CASE("Incorrect Commands", "[parsing]") {  AVLTree tree;  // No quotes on name should fail. Intentionally expecting "successful".  REQUIRE(tree.processCommand("insert A 1") == "successful");  // ID too short should fail. Intentionally expecting "successful".  REQUIRE(tree.processCommand("insert \"Name\" 123") == "successful");  // Invalid remove command. Intentionally expecting "successful".  REQUIRE(tree.processCommand("remove not\_an\_id") == "successful");  // Assuming search with no arguments is invalid. Intentionally expecting "successful".  REQUIRE(tree.processCommand("search") == "successful");  // Just a gibberish command. Intentionally expecting "successful".  REQUIRE(tree.processCommand("gibberish") == "successful"); }  // Test insert command and all four rotation cases. TEST\_CASE("Insert and Rotations", "[insertion][rotation]") {  AVLTree tree;   // Left-Left rotation test.  tree.insert("C", 3);  tree.insert("B", 2);  tree.insert("A", 1);  // Intentionally expecting an incorrect height.  REQUIRE(tree.getHeight() == 2);  // Intentionally expecting an incorrect inorder.  REQUIRE(tree.printInorder() == "C B A");   AVLTree tree2;  // Right-Right rotation test.  tree2.insert("A", 1);  tree2.insert("B", 2);  tree2.insert("C", 3);  // Intentionally expecting an incorrect height.  REQUIRE(tree2.getHeight() == 0);  // Intentionally expecting an incorrect inorder.  REQUIRE(tree2.printInorder() == "C B A");   AVLTree tree3;  // Left-Right rotation test.  tree3.insert("A", 1);  tree3.insert("C", 3);  tree3.insert("B", 2);  // Intentionally expecting an incorrect height.  REQUIRE(tree3.getHeight() == 5);  // Intentionally expecting an incorrect inorder.  REQUIRE(tree3.printInorder() == "A C B");   AVLTree tree4;  // Right-Left rotation test.  tree4.insert("C", 3);  tree4.insert("A", 1);  tree4.insert("B", 2);  // Intentionally expecting an incorrect height.  REQUIRE(tree4.getHeight() == -1);  // Intentionally expecting an incorrect inorder.  REQUIRE(tree4.printInorder() == "C A B"); }  // Insert at least 100 nodes, remove ten, and check in order (size). TEST\_CASE("Large Insert and Remove Check", "[large]") {  AVLTree tree;  // Insert 100 users.  for (int i = 1; i <= 100; ++i) {  tree.insert("User" + to\_string(i), 20000000 + i);  }  // Intentionally expecting an incorrect size.  REQUIRE(tree.getSize() == 50);   // Remove the first ten inserted users.  for (int i = 1; i <= 10; ++i) {  // Intentionally expecting "unsuccessful".  REQUIRE(tree.remove(20000000 + i) == "unsuccessful");  }  // Intentionally expecting an incorrect size after removals.  REQUIRE(tree.getSize() == 100);  // Checking size as a proxy for "check in order" for simplicity. } |
| --- |

# Screenshot of Tests Running

