**B-Tree Test Case Outline**

Capstone Project CS 4488– Andreas Kramer

1. **Test for Insertion:**

* Basic Insertion Test: Insert a few keys and check if they are correctly placed in the tree.
* Insertion Causing Split: Insert enough keys to cause a split in a node and verify if the split is handled correctly.
* Repeated Insertions: Insert a large number of keys and ensure the tree remains balanced.

1. **Test for Searching:**

* Search Existing Keys: After insertion, search for those keys and check if they are found.
* Search Non-existing Keys: Search for keys that aren’t in the tree and ensure the search returns an appropriate response (like null or false).

1. **Test for Deletion:**

* Delete Leaf Node Key: Delete a key from a leaf node and check if it's removed properly.
* Delete Internal Node Key: Delete a key from a non-leaf node and verify the tree restructures correctly.
* Deletion Causing Merge: Delete keys to cause underflow and trigger a merge, then check if merging is correct.
* Deletion Causing Redistribution: Delete keys to cause underflow that leads to redistribution with a sibling node.

1. **Test for Tree Structure and Properties:**

* Tree Height: Verify that the tree height is consistent with the expected height for a B-Tree with its number of keys.
* Balanced Structure: After multiple insertions and deletions, check if the tree remains balanced.
* Correct Ordering of Keys: Ensure that keys in every node are in sorted order, and child pointers are correctly placed.

1. **Stress Testing:**

* Large Number of Operations: Perform a large number of insertions, searches, and deletions to test the tree's performance under load.
* Randomized Operations: Randomly perform a mix of insertions, deletions, and searches to simulate real-world usage.

1. **Edge Case Testing:**

* Empty Tree Operations: Attempt searches and deletions on an empty tree.
* Single Node Operations: Test operations when the tree has only one node.
* Duplicate Key Insertions: Try inserting duplicate keys if your implementation should handle them, and verify the outcome.

1. **Utility Function Tests:**

* Tree Traversal: Verify that in-order, pre-order, or post-order traversals return the correct sequence of keys.
* Print or Debug Functions: Check if utility functions like tree printing or debugging provide the correct representation of the tree.