TEST CASES

TREAPS:

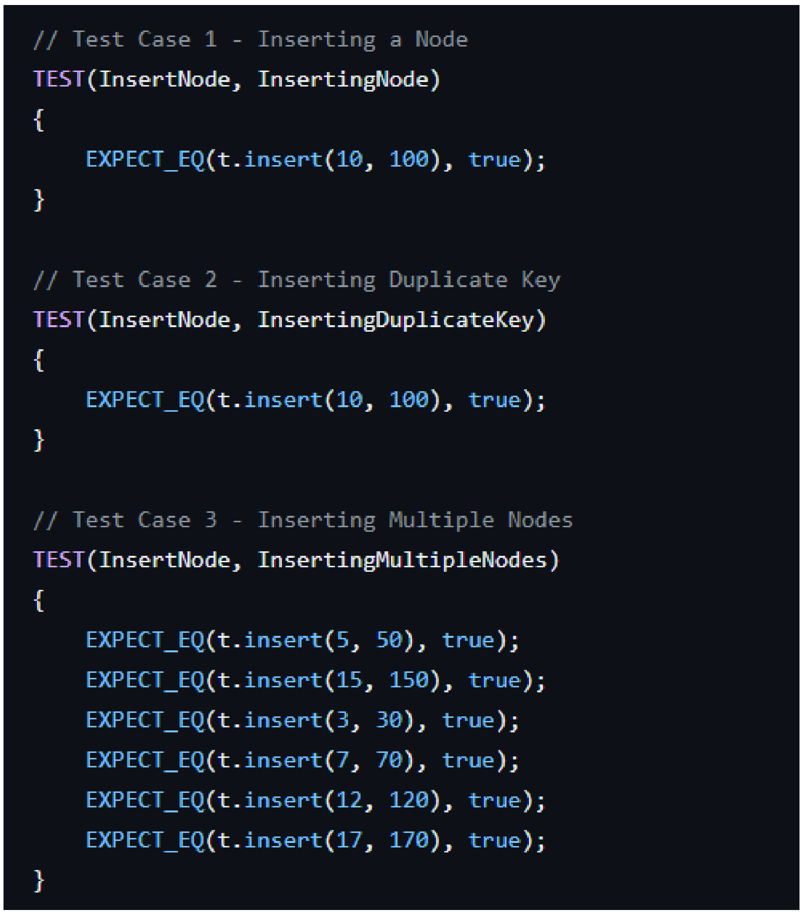
A. INSERTION:

Test Case 1 - Inserting a Node:
Input: Key = 10, Data = 100
Expected Output: Node with key 10 and data 100 should be successfully inserted into the Treap.
Passed in attempt 1

Test Case 2 - Inserting Duplicate Key:
Input: Key = 10, Data = 200 (Duplicate key)
Expected Output: The function should handle duplicate keys appropriately, ensuring the Treap remains balanced.
Passed in attempt 1

Test Case 3 - Inserting Multiple Nodes:
Input: Insert nodes with keys 5, 15, 3, 7, 12, 17
Expected Output: The Treap should maintain the heap property after each insertion.
Passed in attempt 1

CODE:

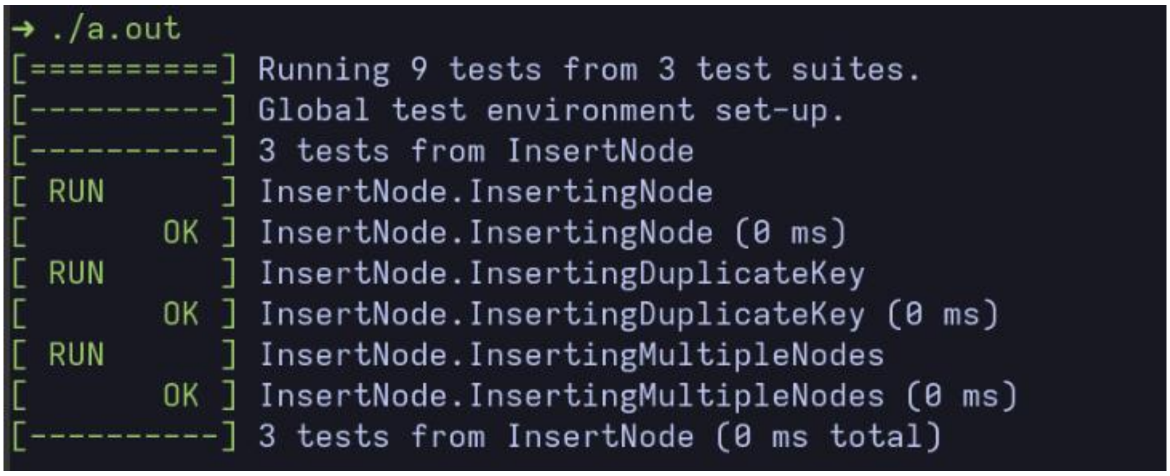


```
// Test Case 1 - Inserting a Node
TEST(InsertNode, InsertingNode)
{
    EXPECT_EQ(t.insert(10, 100), true);
}

// Test Case 2 - Inserting Duplicate Key
TEST(InsertNode, InsertingDuplicateKey)
{
    EXPECT_EQ(t.insert(10, 100), true);
}

// Test Case 3 - Inserting Multiple Nodes
TEST(InsertNode, InsertingMultipleNodes)
{
    EXPECT_EQ(t.insert(5, 50), true);
    EXPECT_EQ(t.insert(15, 150), true);
    EXPECT_EQ(t.insert(3, 30), true);
    EXPECT_EQ(t.insert(7, 70), true);
    EXPECT_EQ(t.insert(12, 120), true);
    EXPECT_EQ(t.insert(17, 170), true);
}
```

RESULT:



```
➜ ./a.out
[==========] Running 9 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 3 tests from InsertNode
[ RUN      ] InsertNode.InsertingNode
[       OK ] InsertNode.InsertingNode (0 ms)
[ RUN      ] InsertNode.InsertingDuplicateKey
[       OK ] InsertNode.InsertingDuplicateKey (0 ms)
[ RUN      ] InsertNode.InsertingMultipleNodes
[       OK ] InsertNode.InsertingMultipleNodes (0 ms)
[----------] 3 tests from InsertNode (0 ms total)
```

B. DELETION:

Test Case 1 - Deleting a Leaf Node:

Input: Key to delete = 3 (a leaf node)
Expected Output: Node with key 3 should be successfully deleted, and the Treap should remain balanced.
Passed in attempt 1

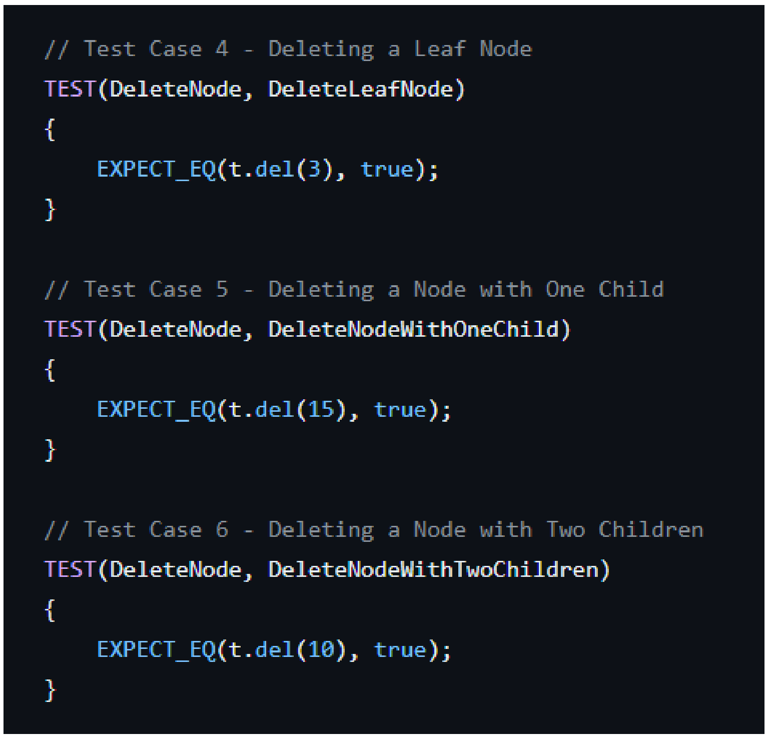
Test Case 2 - Deleting a Node with One Child:

Input: Key to delete = 15 (node with one child)
Expected Output: Node with key 15 should be deleted, and the child should take its place in the Treap.
Passed in attempt 1

Test Case 3 - Deleting a Node with Two Children:

Input: Key to delete = 10 (node with two children)
Expected Output: Node with key 10 should be deleted, and the Treap should be rebalanced while maintaining the heap property.
Passed in attempt 1

CODE:

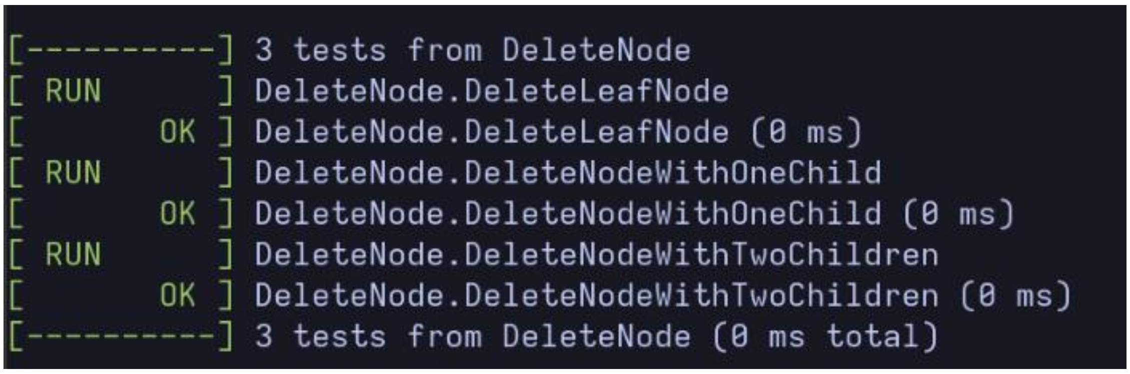


```
// Test Case 4 - Deleting a Leaf Node
TEST(DeleteNode, DeleteLeafNode)
{
    EXPECT_EQ(t.del(3), true);
}

// Test Case 5 - Deleting a Node with One Child
TEST(DeleteNode, DeleteNodeWithOneChild)
{
    EXPECT_EQ(t.del(15), true);
}

// Test Case 6 - Deleting a Node with Two Children
TEST(DeleteNode, DeleteNodeWithTwoChildren)
{
    EXPECT_EQ(t.del(10), true);
}
```

RESULT:



```
[----------] 3 tests from DeleteNode
[ RUN      ] DeleteNode.DeleteLeafNode
[       OK ] DeleteNode.DeleteLeafNode (0 ms)
[ RUN      ] DeleteNode.DeleteNodeWithOneChild
[       OK ] DeleteNode.DeleteNodeWithOneChild (0 ms)
[ RUN      ] DeleteNode.DeleteNodeWithTwoChildren
[       OK ] DeleteNode.DeleteNodeWithTwoChildren (0 ms)
[----------] 3 tests from DeleteNode (0 ms total)
```

C. SEARCH:

Test Case 1 - Searching for an Existing Key:

Input: Key to search = 7 (existing key)
Expected Output: The function should return 1, indicating that the key 7 is present in the Treap.
Passed in attempt 1

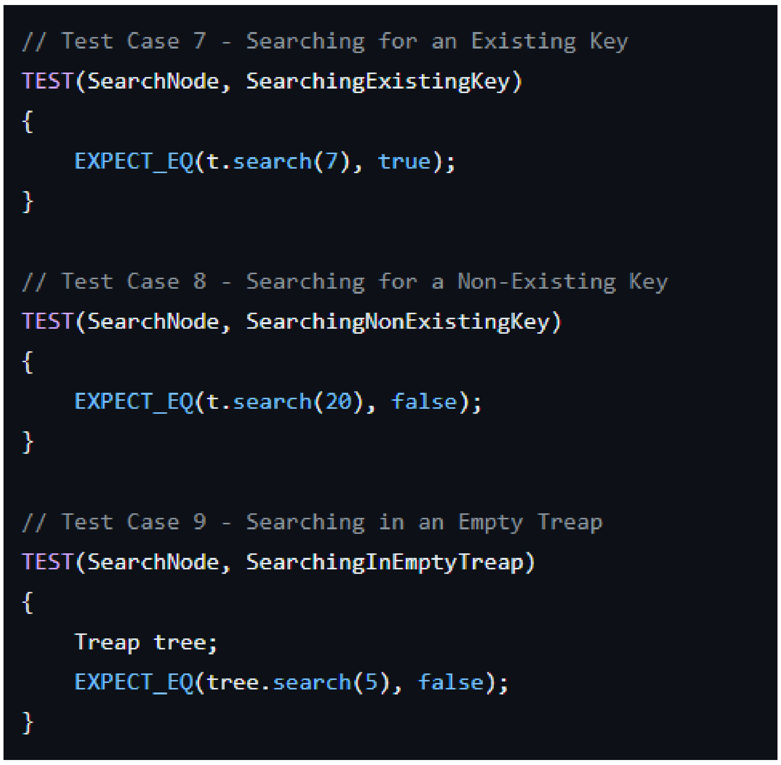
Test Case 2 - Searching for a Non-Existing Key:

Input: Key to search = 20 (non-existing key)
Expected Output: The function should return 0, indicating that the key 20 is not present in the Treap.
Passed in attempt 1

Test Case 3 - Searching in an Empty Treap:

Input: Key to search = 5 (in an empty Treap)
Expected Output: The function should return 0, as the Treap is empty and cannot contain the key 5.
Passed in attempt 1

CODE:

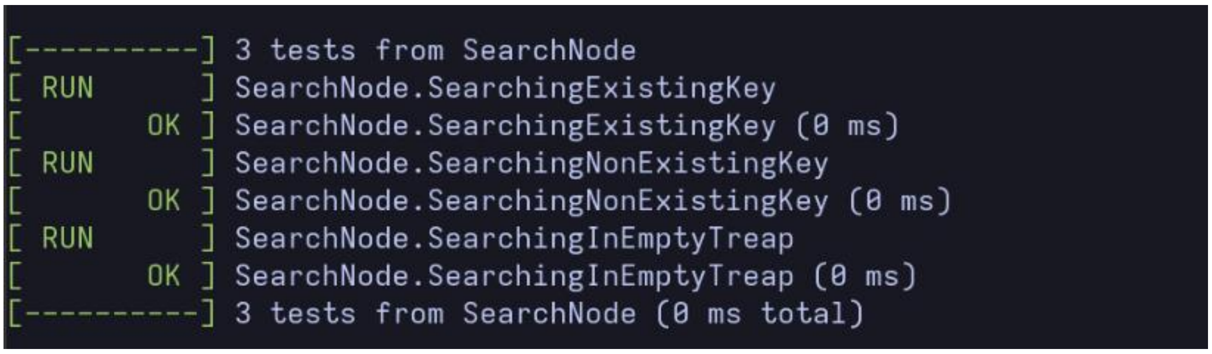


```
// Test Case 7 - Searching for an Existing Key
TEST(SearchNode, SearchingExistingKey)
{
    EXPECT_EQ(t.search(7), true);
}

// Test Case 8 - Searching for a Non-Existing Key
TEST(SearchNode, SearchingNonExistingKey)
{
    EXPECT_EQ(t.search(20), false);
}

// Test Case 9 - Searching in an Empty Treap
TEST(SearchNode, SearchingInEmptyTreap)
{
    Treap tree;
    EXPECT_EQ(tree.search(5), false);
}
```

RESULT:



```
[----------] 3 tests from SearchNode
[ RUN      ] SearchNode.SearchingExistingKey
[       OK ] SearchNode.SearchingExistingKey (0 ms)
[ RUN      ] SearchNode.SearchingNonExistingKey
[       OK ] SearchNode.SearchingNonExistingKey (0 ms)
[ RUN      ] SearchNode.SearchingInEmptyTreap
[       OK ] SearchNode.SearchingInEmptyTreap (0 ms)
[----------] 3 tests from SearchNode (0 ms total)
```