Problem 3 Test Plan

Part a: Y

Part b: Y

Part c: Y

Part d: Y

Part e: Y

The expected output and the actual output match up.

|  |  |  |
| --- | --- | --- |
| Test input | Expected result | Actual result |
| // Create map for int values  map<int> test1;  int a1 = 5;  int a2 = 20;  int b1 = 10;  int c1 = 15;  // test insertion  test1.set("Adam", a1);  test1.set("Bill", b1);  test1.set("Chuck", c1);  test1.set("Adam", a2); // update the existing key value  // display map  test1.display();  // test size  cout << "Size: " << test1.size() << endl;  // test count  cout << "Count for 'Adam': " << test1.count("Adam") << endl;  cout << "Count for 'David': " << test1.count("David") << endl;  // test erase  test1.erase("Bill");  cout << "After erasing 'Bill':" << endl;  test1.display();  // test key\_comparison based on lexicographical order  map<int>::KeyCompare comparison = test1.key\_compare(); // declare comparison object  string key1 = "Adam";  string key2 = "Chuck";  // true condition  if (comparison(key1, key2)) {  cout << "Adam is less than Chuck" << endl;  }  else {  cout << "Adam is not less than Chuck" << endl;  }  // false condition  if (comparison(key2, key1)) {  cout << "Chuck is less than Adam" << endl;  }  else {  cout << "Chuck is not less than Adam" << endl;  }  // try to erase a key that does not exist  try {  test1.erase("Bill");  }  catch (const out\_of\_range& e) {  cout << e.what() << std::endl;  }  // test overload operators  // test []  cout << "Test Overload of the [] operator, ['Adam']: " << test1["Adam"] << endl;  // test ==  map<int> test2;  test2.set("Adam", a2);  // unequal map  bool attempt; // variable to hold the return of the == overload  attempt = test1 == test2;  cout << "Does map1 == map2: " << attempt << endl; // should be false  // equal map  test2.set("Chuck", c1);  cout << "Insert missing pair" << endl;  attempt = test1 == test2;  cout << "Does map1 == map2: " << attempt << endl; // should be true  // Test another type  map<char> test3;  test3.set("Adam", 'a');  test3.set("Billy", 'b');  test3.display(); | Bucket 0:  Bucket 1: {Adam: 20}  Bucket 2:  Bucket 3:  Bucket 4: {Chuck: 15}  Bucket 5:  Bucket 6:  Bucket 7: {Bill: 10}  Bucket 8:  Bucket 9:  Size: 3  Count for 'Adam': 1  Count for 'David': 0  After erasing 'Bill':  Bucket 0:  Bucket 1: {Adam: 20}  Bucket 2:  Bucket 3:  Bucket 4: {Chuck: 15}  Bucket 5:  Bucket 6:  Bucket 7:  Bucket 8:  Bucket 9:  Adam is less than Chuck  Chuck is not less than Adam  Key not found: Bill  Test Overload of the [] operator, ['Adam']: 20  Does map1 == map2: 0  Insert missing pair  Does map1 == map2: 1  Bucket 0:  Bucket 1: {Adam: a}  Bucket 2:  Bucket 3:  Bucket 4:  Bucket 5:  Bucket 6:  Bucket 7:  Bucket 8: {Billy: b}  Bucket 9: | Bucket 0:  Bucket 1: {Adam: 20}  Bucket 2:  Bucket 3:  Bucket 4: {Chuck: 15}  Bucket 5:  Bucket 6:  Bucket 7: {Bill: 10}  Bucket 8:  Bucket 9:  Size: 3  Count for 'Adam': 1  Count for 'David': 0  After erasing 'Bill':  Bucket 0:  Bucket 1: {Adam: 20}  Bucket 2:  Bucket 3:  Bucket 4: {Chuck: 15}  Bucket 5:  Bucket 6:  Bucket 7:  Bucket 8:  Bucket 9:  Adam is less than Chuck  Chuck is not less than Adam  Key not found: Bill  Test Overload of the [] operator, ['Adam']: 20  Does map1 == map2: 0  Insert missing pair  Does map1 == map2: 1  Bucket 0:  Bucket 1: {Adam: a}  Bucket 2:  Bucket 3:  Bucket 4:  Bucket 5:  Bucket 6:  Bucket 7:  Bucket 8: {Billy: b}  Bucket 9: |