```
#include <iostream>
#include <string>
#include <gtest/gtest.h>
#include "vector_collection.h"

using namespace std;

// Test 1
TEST(BasicsListTest, CorrectSize) {
 VectorCollection<string,double> c;
 ASSERT_EQ(c.size(), 0);
 c.insert("a", 10.0);
 ASSERT_EQ(c.size(), 1);
 c.insert("b", 20.0);
 ASSERT_EQ(c.size(), 2);
}

// Test 2
TEST(BasicListTest, InsertAndFind) {
 VectorCollection<string,double> c;
 double v;
 ASSERT_EQ(c.find("a", v), false);
 c.insert("a", 10.0);
 ASSERT_EQ(c.find("a", v), true);
 ASSERT_EQ(v, 10.0);
 ASSERT_EQ(c.find("b", v), false);
 c.insert("b", 20.0);
 ASSERT_EQ(c.find("b", v), true);
 ASSERT_EQ(v, 20.0);
 // insert/find a string that's longer than 1 char
 c.insert("GoNzAgA", 22);
 ASSERT_EQ(c.find("GoNzAgA", v), true);

}

 // Test 3
TEST(BasicListTest, RemoveElems) {
 VectorCollection<string,double> c;
 // attempt to remove from an empty object
 c.remove("");
 ASSERT_EQ(c.size(), 0);
 c.insert("a", 10.0);
 c.insert("b", 20.0);
 c.insert("c", 30.0);
 double v;
 c.remove("a");
 ASSERT_EQ(c.find("a", v), false);
 c.remove("b");
 // attempt to remove a string that isn't in vector
 c.remove("h");
```

```
 ASSERT_EQ(c.find("b", v), false);
 c.remove("c");
 ASSERT_EQ(c.find("c", v), false);
 ASSERT_EQ(c.size(), 0);
}

 // Test 4
TEST(BasicListTest, GetKeys) {
 VectorCollection<string,double> c ;
 c.insert("a", 10.0);
 c.insert("b", 20.0);
 c.insert("c", 30.0);
 vector<string> ks;
 c.keys(ks);
 vector<string>::iterator iter;
 iter = find(ks.begin(), ks.end(), "a");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "b");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "c");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "d");
 ASSERT_EQ(iter, ks.end());
}

 // Test 5
TEST(BasicListTest, GetKeyRange) {
 VectorCollection<string,double> c;
 c.insert("a", 10.0);
 c.insert("b", 20.0);
 c.insert("c", 30.0);
 c.insert("d", 40.0);
 c .insert("e", 50.0);
 vector<string> ks;
 c.find("b", "d", ks);
 vector<string>::iterator iter;
 iter = find(ks.begin(), ks.end(), "b");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "c");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "d");
 ASSERT_NE(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "a");
 ASSERT_EQ(iter, ks.end());
 iter = find(ks.begin(), ks.end(), "e");
 ASSERT_EQ(iter, ks.end());
 // test if find range works for strings
 // longer than one character
 VectorCollection<string,double> d;
 d.insert("apples", 10);
```

```
 d.insert("zebras", 20);
 d.insert("golf", 30);
 d.insert("hydro", 40);
 vector<string> ks2;
 d.find("bees", "yoyo", ks2);
 vector<string>::iterator iter2;
 iter = find(ks2.begin(), ks2.end(), "golf");
 ASSERT_NE(iter, ks2.end());
 iter = find(ks2.begin(), ks2.end(), "hydro");
 ASSERT_NE(iter, ks2.end());
 iter = find(ks2.begin(), ks2.end(), "apples");
 ASSERT_EQ(iter, ks2.end());
 iter = find(ks2.begin(), ks2.end(), "zebras");
 ASSERT_EQ(iter, ks2.end());
 ASSERT_EQ(ks2.size(), 2);
}

 // Test 6
TEST(BasicListTest, KeySort) {
 VectorCollection<string,double> c;
 c.insert("a", 10.0);
 c.insert("e", 50.0);
 c.insert("c", 30.0);
 c.insert("b", 20.0);
 c.insert("d", 40.0);
 vector<string> sorted_ks;
 c.sort(sorted_ks);
 // check if sort order
 for(int i = 0; i < int(sorted_ks.size()) - 1; ++i)
  ASSERT_LE(sorted_ks[i], sorted_ks[i+1]);

 VectorCollection<string, int> d;
 d.insert("anagrams", 23);
 d.insert("hemmingson", 1);
 d.insert("string", 44);
 d.insert("Regis", 99);
 d.insert("Salem Oregon", 98);
 d.insert("turing", 23);
 vector<string> sorted_stringInt;
 d.sort(sorted_stringInt);
 //check if sort order
 for (int i = 0; i < int(sorted_stringInt.size())-1; ++i)
  ASSERT_LE(sorted_stringInt[i], sorted_stringInt[i+1]);
}

TEST(BasicListTest, Negatives) {
 VectorCollection<double,string> e;
 e.insert(999.0, "DigitalLogic");
 e.insert(400.4, "AlgsAndDataStruct");
 e.insert(0.0, "Human Nature");
```

```
 e.insert(-33.2, "discreteMath");
 e.insert(-0.1, "Globals");
 vector<double> sorted_ints;
 e.sort(sorted_ints);
 for (int i = 0; i < int(sorted_ints.size()-1); ++i)
 {
  ASSERT_LE(sorted_ints[i], sorted_ints[i+1]);
 }
}

TEST(BasicListTest, SizeZero) {
 VectorCollection<int, int> f;
 ASSERT_EQ(f.size(), 0);
 f.remove(2);
 ASSERT_EQ(f.size(), 0);
 int my_val;
 ASSERT_EQ(f.find(0, my_val), false);
 vector<int> keys_ints;
 f.keys(keys_ints);
 ASSERT_EQ(keys_ints.size(), 0);
 f.sort(keys_ints);
 ASSERT_EQ(keys_ints.size(), 0);
}

int main(int argc, char** argv)
{
 testing::InitGoogleTest(&argc, argv);
 return RUN_ALL_TESTS();
}
```