Part 1 Testing Strategy

Paul Nickerson

I attempted to separate tests into two somewhat distinct categories: operations that are expected to fail (operation_failures.cpp), and operations that are expected to succeed (operation_successes.cpp).

operation_failures.cpp

Within the failure operations scenario, I start with an empty map and try to search() and remove() an item whose key does not exist in the map. Both calls should return false. I then fill the map to capacity, which is possible to do since linear probing allows us to reliably fill every slot, and attempt to insert() a key (should fail due to lack of space), remove() a key that doesn't exist in the map, and search() for a key that does not exist in the map.

operation_successes.cpp

The success-expecting operations is much more extensive. I start by filling the map halfway, clearing it, then filling it up halfway again. The map should then report the correct size. I check that several keys which are expected to exist in the map actually do exist (including the lowest possible key, the highest possible key, and one in the middle).

I check the print() function by routing it to an output string stream and count the number of hyphens in the output, which indicate empty slots. I ensure that these match the value of capacity() - size().

I then attempt to remove() several keys which are known to exist, and check that their associated values are what were expected. After these items are removed, I try to both search() and remove() them, which should all return false.