Part 2 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 with a bunch of items (it is impossible to run out of space because collisions are resolved via an arbitrarily-growable liked list). From this newly-filled map, I attempt to remove() a key that doesn't exist in the map, and search() for a key that does not exist in the map, both of which should return false.

operation_successes.cpp

The success-expecting operations is much more extensive. I start by filling the map with a bunch of items, clearing it, then filling it up 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. Since load factor = occupied buckets / capacity, we can get the number of unoccupied buckets as capacity * (1 - load). This should equal the number of hyphens in the print() output.

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.