Testing Strategy for Part 3: Bucket

I started out with a hashmap of size 20 that used ints and characters. I first tested insert and the replacement part of insert by inserting two values into the same key and then searching to find the value of the key, making sure that it was the second value inserted and also testing the number of probes required against my known values. I then made sure that removes and searches properly returned the number of probes times negative one when the value was not in the hashmap and did not affect the values within it. Then, because this is a hash bucket I inserted many keys that all mapped to the same value to ensure they simply got added on to the linked list and still left other slots open and that load() properly accounted for this.

Using the hashmap created from this sequence of searches and removes, I tested the clear, is_empty, print, load, and size functions. I then insert many values into the hashmap and tried a sequence of inserts, removes, and searches to again ensure its validity.

Then, I tested the cluster_distribution functions and remove_random functions by seeing the cluster distribution after the print statement to ensure its validity then randomly removing a large number of keys and printing which keys were removed and running through this multiple times to ensure they were randomly chosen. Then, I again printed and used cluster_distribution to ensure that they were working properly.

Next, I started out with a hashmap of strings and integers and inserted, searched, removed, printed, etc. and ensured they were valid. I did the same with C-Strings and chars and doubles and ints to test the templating and ensure that all types of keys were supported.

I chose hashmaps with sizes multiples of 10 and a poorly chosen hash function for maps of this size so it was easier to determine correct behavior because it was obvious where each key would be added.