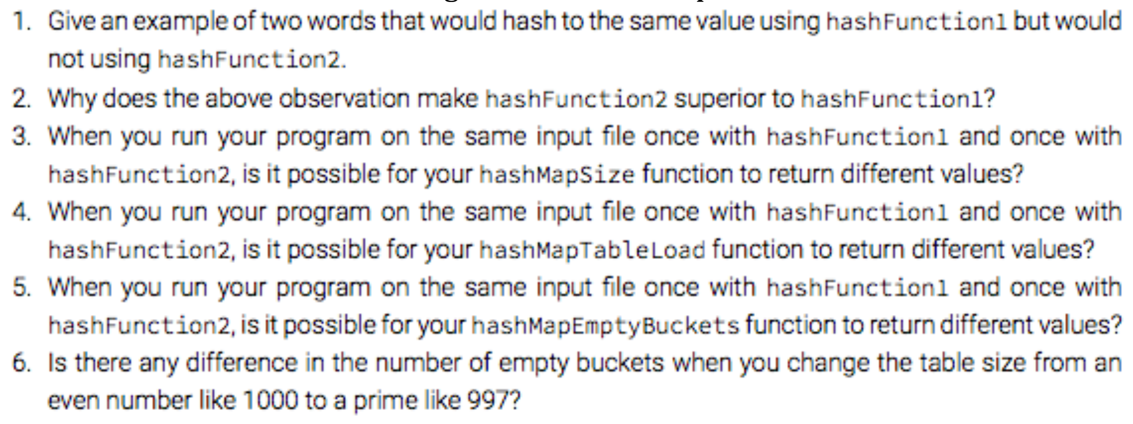
Jacob Karcz

Assignment 6, written part



1. Give an example of two words that would hash to the same value using `hashFunction1` but would not using `hashFunction2`.
2. Why does the above observation make `hashFunction2` superior to `hashFunction1`?
3. When you run your program on the same input file once with `hashFunction1` and once with `hashFunction2`, is it possible for your `hashMapSize` function to return different values?
4. When you run your program on the same input file once with `hashFunction1` and once with `hashFunction2`, is it possible for your `hashMapTableLoad` function to return different values?
5. When you run your program on the same input file once with `hashFunction1` and once with `hashFunction2`, is it possible for your `hashMapEmptyBuckets` function to return different values?
6. Is there any difference in the number of empty buckets when you change the table size from an even number like 1000 to a prime like 997?

1. {eat, ate, tea} {bin, nib} {diem, dime} {made, dame} {bean, bane) {vein, vine}

2. HashFunction2 is superior because it returns a unique numeric value for each letter of the key based on the letter and its position whereas function 1 simply adds the ascii value of the letters in the key, so words like tea, ate and eat all hash to the same value even though they are all unique and acceptable. So hashFunction 1 would lead to a lot more collisions and much less efficient data searches leading to a less desirable big O complexity for a hashMap implemented with hashFunction1 than a hashMap implemented with hashFunction2 with the same total table load value (ie if the table load is .2 but all the values are in one array index, that's a O(n) while the same load with the associations stored at separate indexes is a O(1).

3. No, we are still adding the same number of items to both hashMaps, they are just hashing to a different array index. So regardless of the function, adding x number of elements to an empty map will yield a map of size x.

4. No, as was the case in question 3, the capacity and size of the maps will be the same and the capacity will double when the same number of elements are added and compared to the desired maximum load. The hash function will have no effect on when the size of the map reaches critical capacity proportion, and thus the load will return the same value with a poor hash function as well as a perfect hash function.

5. Yes, it would be almost expected that the number of empty buckets would be different for each function, as one will hash more efficiently and lead to less collisions while the other will not be as effective and place more links in the same bucket (leading to more empty buckets with hashfunction1).

6. Yes, prime numbers are much more effective in ensuring that new items hash to different buckets, this leads to a more even distribution of elements and a more efficient search algorithm (in the contains, put, remove, and get functions).