```
//*********************************************************************************
//
//     CS261- Assignment 5 - Written
//
//     Name:         Tucker Dane Walker
//     Date:         10 August 2017
//     Description:  Written Answers to Questions on Assignment 5
//
*********************************************************************************/
```

1. Give an example of two words that would hash to the same value using hashFunction1 but would not using hashFunction2

   "ate", "*eat"*, and "*tea"* would hash to the same value using hashFunction1 because hashFunction1 simply adds up the values of each letter in the word. In this case, all letters in these words are the same. hashFunction2, on the other hand, modifies the hash value based both on the values of the characters in the string and their position.

2. Why does the above observation make hashFuncion2 superior to hashFunction1?

   The above observation makes hashFunction2 superior to hashFunction1 because it avoids clustering of values with the same characters within the same buckets; it spreads them out across the hash table. This is more likely to make accessing these values faster because strings containing the same characters will not likely be in a linked list together.

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?

   No. hashMapSize will return the number of links in the hashMap. The number of links is equal to the number of unique strings in the file. If the strings in the file does not change, then the hashMapSize will not change.

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?

   No. hashMapTableLoad refers to the size of the map divided by its capacity. These values do not change with the hash function used. Clustering is more/less likely to occur with different hash functions, but neither the size nor the capacity will change. Therefore, the load will not change.

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?

   Yes. The hash functions control what buckets the values are eventually stored in. This is because they produce the key to which each value is stored. When running hashFunction1 on input3.txt, clustering occurs and values are stored in the same bucket: *"ate"*, *"eat"*, and "*tea"* are all stored in bucket 4. While running hashFunction2 on the same input file, these strings are all stored in three different buckets. The total number of buckets did not change between the two iterations. Therefore, the first has more empty buckets than the second.

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?

Yes/No. There *can* be a difference in the number of empty buckets but not in every case. This depends on a number of factors: The keys being passed and the hash function used. Obviously, in the example listed above, there are three fewer buckets overall, so this too may affect the number of empty buckets. That being said, generally speaking, the number of empty buckets *is more likely to change* as the table size is changed to a prime number because 1. The number of buckets changes and 2. Unique bucket locations are more likely to be generated due to the inherent uniqueness of the prime number (prime numbers being unique because they are not divisible by any other number). Therefore, not only are the number of empty buckets more likely to change by using a prime number for the table size, but there is a likelihood that there will be less clustering of key/value pairs in the same buckets will occur (so long as the prime number selected is relatively close in value to the original non-prime number, as in 997 and 1000).