1. Give an example of 2 words that would hash to the same value using stringHash1() but would not using stringHash2().
   1. Based on the code, stringHash1() simply takes the ASCII values of the string and returns them. Because of this, any words that use the same letters will return the same value for it (IE “on” and “no”). stringHash2() instead multiplies the ASCII value by the position.
2. Why does the above make stringHash2() superior to stringHash1()?
   1. As said above, stringHash1() will give the same result as long as the same letters are used. stringHash2() will instead give a unique value for each possible word.
3. When you run your program on the same input file but one run using stringHash1() and on the other run using stringHash2(). Is it possible for your size() function to return different values?
   1. I believe the size function will always be the same no matter which hash function you use because it just returns the number of hashlinks in the table.
4. When you run your program on the same input file using stringHash1() on one run and using stringHash2() on another, is it possible for your tableLoad() function to return different values?
   1. Based on my answer for number 3 (which I have concluded is the same no matter which version is used), this function just divides that by the total number of buckets meaning it should be the same (as long as my conclusion about question3 is correct).
5. When you run your program on the same input file with one run using stringHash1() and the other run using stringHash2(), is it possible for your emptyBuckets() function to return different values?
   1. To be honest I am not quite sure. I checked if a specified value was NULL and considered that an emptyBucket; in which case I suppose it could be possible as the key values could be the same (stringHash1()) and just assign different values for a specified key (thus increasing the number of empty buckets)
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. At least the way I coded it, I don’t believe there is any unknown circumstances (except of course the -3 in total buckets). I just went through the table and counted the number of empty buckets.
7. Using the timing code provided to you, run your code on different size hash tables. How does affecting the hash table size change your performance?
   1. Simply changing the tableSize variable in main.c hasn’t lead to any considerable results. I increased from 10 to 100 in 1 test and both seemed to yield between 0.0002 and 0.0003 seconds.