1. The words "tea" and "eat" would hash to the same value when using hashFunction1 but not when using hashFunction2 because hashFunction2 accounts for the ordering of the letters.

2. The second hash function is superior to the first hash function because it accounts for char index position (the ordering of the chars) as well as their corresponding ASCII value which better minimizes the number of links assigned to the same bucket which helps the hash table to maintain O(1) link lookup.

3. It is not possible for the hashMapSize function to return different values when the same input file is used because the same words would be inputted which would result in the same total number of HashLinks.

4. It is not possible for the hashMapTableLoad function to return different values when the same input file is used because the hashMapTableLoad function simply finds the ratio between the number of links (unique words) and the total number of buckets. The function does not track how many of the buckets are empty and the tables in both cases will resize at the same rate so number of unique words and total number of buckets will remain the same independent of the hashing function used.

5. Yes, it is possible for the hashMapEmptyBuckets method to return different values because the different hashFunctions will result in different numbers of collisions and therefore one hashTable will result in more empty buckets than the other even with the same initial table size.

6. Yes, there is a difference in the number of empty buckets when the initial hash table size is set to an even number such as 1000 verses when the initial hash table size is set to a prime number such as 997. Because table capacity is used in the modulo operation to determine the bucket placement, a prime number capacity decreases the chances of a collision because it only has two common factors by definition (itself and one) while composite numbers can have more than two.