1. eat, tea

2. stringHash2() will have fewer collisions. Anytime there is a collision the new word will be placed in the same linked list bucket. Lowering the number of collisions would keep the length of each linked list short and minimize the search time.

3. No, the number of hash links will remain the same since a new hash link is added for each new word.

4. No, since the number of hash links is the same for both the table load will also be the same.

5. yes, for the example words: tea, eat, ate, will all hash to one bucket with stringHash1() but 3 separate buckets with stringHash2(). This would create a different number of empty buckets.

6. For stringHash1() there was a small decrease in the number of empty buckets (6 buckets). For stringHash2() there was a small increase in the number of empty buckets (15 buckets). Since that seems fairly inconclusive I ran it again with a table size of 5 and 4. I don't know if I just found a special case but for both hash functions there was a large decrease in empty buckets for the prime (5) table starting point by almost half. My explanation for this is that since the table size is prime and we are taking the mod of the hash value and the table size, there should be less collisions. This is because there are no common factors for the prime. This should spread the table out more and create fewer buckets.

7. I ran each hash function with several table sizes (see data below). For hash1 it looks like the run time increases as the table size increases. For hash2 it seems to roughly follow that trend if you ignore the first result. I reran hash2 with table size 1 a second time and got 6.340 sec so the run time seems to depend on something other than the table size. Probably whatever else is running on the computer at the same time.

Hash 1	Text #	empty Buckets	table count	capacity	load	table start
0.153	1	40	26	64	0.40625	1
5.43	2	1566	908	2048	0.443359	1
6.996	2	799	908	1280	0.709375	5
6.497	2	1566	908	2048	0.443359	4
7.328	2	1512	908	1994	0.455366	997
7.03	2	1518	908	2000	0.454	1000
Hash 2						
0.157	1	45	26	64	0.40625	1
7.482	2	1430	908	2048	0.443359	1
6.721	2	720	908	1280	0.709375	5
7.086	2	1430	908	2048	0.443359	4
7.203	2	1406	908	1994	0.455366	997
7.289	2	1391	908	2000	0.454	1000