1) Algorithmic asymptotic complexity:
   a) Hash function: O(n) – linear. This function uses a single loop that iterates once for each character in the string. As more characters are added to a word, the complexity grows linearly.
   b) Search Value: O(1) – linear. Although this function should grow with linear complexity because there is a loop in the implementation, the rest of the hash table is set up so that the the size of the bucket array is larger than twice the number of elements. This means there will never be a collision. So, searching a value should be instantaneous.
2) Properties of a good hash function:
   a) Resizes size of bucket array before the number of elements is equal to half the size of the current size. This not only minimizes collisions, but eliminates them altogether, as discussed in lecture.
   b) Should place elements as far apart from each other as possible to prevent clustering. Although collisions shouldn't be an issue anyways, for reasons discussed above, but minimizing clustering is another good way to prevent collisions.
   c) The hash function uses all the hash data. If the hash function only uses certain parts of the data for hashing, then two distinct elements may get hashed to the same location. In fact, part (a) only holds true if this is also true.
   d) The hash value is totally determined by the data being hashed, rather than using outside elements. If outside elements are used for hashing, discrepancies in these outside elements could lead to collisions.
   e) The hash function yields very different results for similar inputs. In real world applications, similar inputs are likely to be passed into a hash table. If these are hashed to very different locations, this is a sign that the hash function is doing a good job at separating data evenly, which is important for reasons discussed in part (b).
3) My hash function takes the ascii value of each character in the input string, sums them together, and modulates it with respect to the size of the table. My hash function does not do a good job separating similar data entries evenly across the table. In fact, sample inputs: "aa, ab, ac, etc." will be placed right next to each other on the table. Furthermore, it does an even worse job with values such as "ac" and "ca", which will be hashed to the same spot on the table. This is obvious, because they have the same ascii values summed together, but there are also less obvious collisions. For instance, my hash function hashed the words "forgotten" and "prominent" to the same location.