# CS2302 Data Structures
## Spring 2020
### Quiz – Hash Tables with Chaining

Use the hash table that solves collisions by chaining implemented in the program *hash_table_chain.py* provided in the class blackboard page to solve efficiently each of the following problems:

1. Write the function *location(h,k)* that receives a hash table *h* and an integer *k* and returns the index of the bucket and the index within that bucket where the record with key *k* appears. If there is no record with key *k*, your function should return the index of the bucket where *k* would be if *k* were in the table and *None* as the index value (see expected results in starter code).

2. Write the function *change_key(h,k,new_k)* that receives a hash table *h* and integers *k* and *new_k* and modifies the record with key *k* in *h* to have a key value of *new_k*. However, if *k* is not in the table, or *new_k* is already in the table, your function should do nothing. Notice that changing the key will likely mean that the record needs to be moved to a different bucket (see expected results in starter code).

3. Write the function *unique_items(L)* that receives a list of integers *L* and returns a list containing the items that appear at least once in *L*, sorted in ascending order (you may use the built-in function x.sort() that sorts list x; see expected results in starter code). This should run in *O(n)* time.

4. (Extra credit 1, do not use hashing) Write the function *sum2(L,k)* that receives a list of integers *L* and an integer *k* and returns two indices *i* and *j* such that L[i] + L[j] = k. If there are no such indices, your function should return *None, None*. Write the first version of this function that runs in $O(n^2)$ time by exhaustively trying all pairs of elements in *L*.

5. (Extra credit 2, use hashing) Write the function *sum2HT(L,k)* just like in the previous question but the function must now use a hash table and run in *O(n)* time.