Lince Rumainum
Student ID: 112722337
CS 2413 Data Structures
Project 3

For Project 3, one of alternate hash function options would be a mid-square method hash function. This hash function will take the key (in project 3, that would the record ID), calculate the square of it, and then take the middle digits (i.e. 2 digits or convert it to binary and take however many middle binary digits specified in the function) as the hash value. Using this hash function will change the code complexity. It will take a longer time to compute (insert, search, or remove) due to more calculation being made and more storage space needed to store the square of that number and the middle number of that result. The code will also become more complex since it will need to determine the middle position of the square result of each key. The efficiency of the mid-square method hash function is good since the data will most likely to be spread uniformly since collisions will least likely to happen. The code efficiency using the required hash function could either be good or bad depend on the data. Since it is a division modulo, the hash function is very fast to compute. The reminder of the key divide by the hash table size become the hash value. The problem will occur if the data create a lot of collisions (i.e. if the hash table size is 11 and keys are multiples of 11, the hash value will all be at bucket 0 and collisions will occur every time user try to insert data. It will also take Big O(n) to search and remove data).

One of alternate hash collision resolution strategies would be a linear probing. Linear probing will resolve collision by linearly find the closest empty bucket position from where the collision happened. The time complexity of linear probing to insert, search, or remove data is Big O(n) if all the data present have the same hash value. Since finding and removing data from the hash table with linear probing is to check if the bucket is empty or not, the space complexity of the linear probing would be Big O(n). The code efficiency of a hash table with 0.9 load factor using linear probing would have an average of 50.50 probes for an unsuccessful search and 5.50 probes for a successful search while the required collision resolution strategy (separate chaining strategy) has an average of 1.31 probes for an unsuccessful search and 1.45 probes for a successful search. The numbers show separate chaining is a better hash collision resolution strategy especially when handling a significant amount of data.