Assignment 5. **Hashing and hash functions.**
Given February 28, due March 7.

1. We have a large collection of short quotes and sayings. The time to do an equality test between two quotes is proportional to the number of characters (bytes) in the quote, which would typically be on the order of fifty. The time to compute a 32 bit integer hash function for such a quote is also proportional to the number of characters. These integers will be called “big hash keys” because they come from a hash function but we don’t imagine using a hash table big enough to hold all $2^{32}$ possible addresses. Right now, the quotes are in no particular order and have come from many sources. We want to create a database of quotes so that we can easily remove duplicates. We think about 10% are duplicates. Compare the following four strategies and recommend one of them. Make your reasoning quantitative, making realistic estimates of the relative running times.

   a. Sort the quotes, say, using quicksort and then make a pass to remove duplicates.

   b. Sort the big hash keys and make a pass to remove duplicate keys. Even though there is only a small chance that the quotes could be different given that the big hash keys are the same, we should probably do an equality test for the quotes themselves before declaring them to be duplicates. Is it reasonable to think that we can compare big hash keys faster than quotes?

   c. Hash the big hash keys into a hash table whose size is roughly the size of the number of quotes and let hashing find the duplicate big hash keys.

   d. Never mind the big hash keys, just hash the quotes into a table maybe twice the size of the number of quotes.

2. Read CLR section 6.6 paying particular attention to the formula at the top of page 129. Suppose we have $n = 10^6$ (a million) quotes and a perfect hash function to compute the 32 bit big hash keys. Do we expect that there is a pair of quotes that are different but have the same big hash key? If so, roughly how many times will that happen? If there are 10% true duplicates, what is the ratio of the number of true duplicates to false duplicates. A pair of quotes is a false duplicate if their big hash keys agree even though the quotes are different. Use this to comment on the slowdown coming from the need to test for false duplicates.

3. We have a $N$ (very large number of) sales records. Each record consists of the id number of the customer and the price. There are $k$ customers, where $k$ is still large, but not nearly as large as $N$. We want create a list of customers together with the total amount spent by each customer. That is, for each customer id, we want to know the sum of all the prices in sales records with that id. Suggest a sensible algorithm for doing this. Give details such as hash table sizes but assume the existence of hash functions.