Samip Shah
samipjanish.shah@mavs.uta.edu
UTA ID # 1001709873
CSE5311

**(1) Suppose we use a hash function h to hash n distinct keys into an array T of length m. Assuming simple uniform hashing, what is the expected number of collisions? More precisely, what is the expected cardinality of {{k,l};k != l and h(k) = h(l)} ?**

We will use linearity of expectation to solve this.
Suppose all the keys are pressed in order of k1, k2, k3,…,kn. Now let Xi be number of l>ki, such that h(l) = h(ki). Then by using linearity, number of collisions is sum of total collision for each smallest collision.
Therefore, $\sum_{i=1}^{n}(n-i)/m$ = [$n^2$ – n(n+1)/2]/m = ($n^2$ – n)/2m

**(2) Suppose that we are storing a set of n keys into a hash table of size m. Show that if the keys are drawn from a universe U with |U| > nm, then U has a subset of size n consisting of keys that all hash to the same slot, so that the worst-case searching time for hashing with chaining is Θ(n).**

Lets assume that m-1 solts have been filled with at most n-1 elements into the hash table. Now for just one slot left
|U| - (m - 1)(n - 1)
But |U| > nm thus we can write above equation as
|U| - (m - 1)(n - 1) > nm - (m - 1)(n - 1)
= nm – [nm – m – n + 1]
= n + m – 1 which is always greater than n.
Thus U is subset of n

**(3) Suppose we wish to search a linked list of length n, where each element contains a key k along with a hash value h(k). Each key is a long character string. How might we take advantage of the hash values when searching the list for an element with a given key?**

If every element contains hash of long character string while we are searching for specific element in linked list, then we will check hash value of every node in linked list and move to next node if not found. This increase the run time compared to length of long character string.

Reference: https://walkccc.github.io/CLRS/Chap11/11.2/
https://math.stackexchange.com/questions/2051195/what-is-the-expected-number-of-collisions