Homework 10: Hashing

Data Structures

Write pseudo-code not Java for problems requiring code. You are responsible for the appropriate level of detail.

1. If a hash table contains *tablesize* positions and *n* records currently occupy the table, the load factor *lf* is defined as *n/tablesize.* Suppose a hash function uniformly distributes *n* keys over the *tablesize* positions of the table and the table has load factor *lf*. Show that of new keys inserted into the table, (n-1)\**lf*/2 of them will collide with a previously entered key. Think about the accumulated collisions over a series of collisions.
2. Assume that *n* random positions of a *tablesize*-element hash table are occupied, using hash and rehash functions that are equally likely to produce any index in the table. The hash and rehash functions themselves are not important. The only thing that is important is they will produce any index in the table with equal probability. Start by counting the number of insertions for each item as you go along. Use that to show that the average number of comparisons needed to insert a new element is *(tablesize* + 1*)*/*(tablesize*-n+1). Explain why linear probing does not satisfy this condition.