# CMSC 341 Data Structures
# Hashing Review

October 26, 2010

1. What is a hash function? Name two desirable properties of a hash function.
2. Define **collision** in a hash table.
3. What is the **clustering** problem in hash tables?
4. Describe the **division method** for generating hash values.
5. Describe the **multiplication method** for generating hash values.
6. Define **Fibonacci hashing**.
7. Describe the **separate chaining** collision resolution method.
8. Describe the **open addressing** collision resolution method.
9. Given a hash table of size 13, show the contents of your hash table after inserting the values {8, 2, 7, 18, 15, 19, 13, 23, 15, 20, 16} using open addressing with linear probing
( $f(i) = i$ ) for collision resolution.
10. Repeat question 9, using open addressing with quadratic probing ( $f(i) = i^2$ ) for collision resolution.
11. Repeat question 9 using separate chaining for collision resolution.
12. The average time performance of the insertion and searching operations on a hash table are $O(1)$, which is much better than the performance of a binary search tree for the same operations. Given this wonderful performance of hash tables as compared to binary search trees, when would you want to use a binary search tree instead of a hash table?

13. In a hash table using open addressing with linear probing, the average number of probes for successful search , S, and unsuccessful search (or insertion), U, are

$$S \approx \frac{1}{2}\left( 1 + \frac{1}{1-\lambda} \right)$$

$$U \approx \frac{1}{2}\left( 1 + \frac{1}{(1-\lambda)^2} \right)$$

where $\lambda$ is the load factor of the table.
Suppose you want a hash table that can hold at least 1000 elements. You want successful searches to take no more than 4 probes on average.

(a) What is the maximum load factor you can tolerate in your hash table?

(b) If the table size must be prime, what is the smallest table size you can use?

(c) Based on your answers to (a) and (b), what is the average number of probes to perform an insertion?