[150] **Project 2**. *Hashing Finds Common Words*

Due: April 22 by midnight.

The purpose of this project is to implement a hash function and hashing techniques, and to gain some insight into the management and the performance evaluation of hash tables. The problem to be solved is defined as follows: Given two text files: `text1` and `text2` that can be found on `http://www.cs.purdue.edu/homes/spa/cs251.html`, determine the words that appear in both files. (A word is any sequence of characters including hyphen not containing the blank symbol or punctuation symbol.) The problem should be solved by building a hash table containing all the words in file `text1`, and then performing a search for every word in file `text2`.

1. The Hash Functions

Let $x = x_{k-1} \cdots x_1 x_0$ be a character string of length k and let B be the size of the hash table. Let *bin(x)* be the value of the binary number formed by the string x; i.e., $bin(x) = \sum_{i=0}^{k-1} `x_i`(2^8)^i$, where $`x_i`$ represents an integer value of the character x_i. The hash function $h(x)$ is defined as

$$h(x) = bin(x) \mod B.$$

Before implementing $h(x)$ get a feeling for how large $bin(x)$ will get. In order to avoid overflow and underflow (i.e., loss of significant digits) make use of the equations:

$$(a + b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m \text{ and}$$
$$(a * b) \bmod m = ((a \bmod m) * (b \bmod m)) \bmod m.$$

2. The Hashing Techniques Hash Tables

You are supposed to implement both:

- closed hashing with *quadratic probing*,
- open hashing technique with *chaining*.

Since you do not have accurate estimates on the size of the hash table needed, you have to reconstruct the hash table if it gets too full. You should start with a table of size S1 = 71 and monitor constantly the load factor, which is the ratio of occupied locations (number of stored items) to the total number of positions in the table. Whenever this load factor increases beyond a certain threshold τ, a new, larger, table of roughly twice the size of the old one is constructed, and all the entries in the old table are *rehashed* into the new table. If a table of size 71 gets too full, create one of size 149. This process may be repeated for table sizes S3 = 307, S4 = 617, if needed. Take τ to be 0.95.

3. Output

You should:

- print common words for both files `text1` and `text2` that you found using both open and closed hashing techniques.
- count the number of *distinct* words in each file

- estimate the average number of comparisons (per word) as a function of the load factor. More specifically, during the construction of each hash table (while reading `text1`), you should consider the following ranges of load factors: [0,0.5), [0.5,0.65), [0.65,0.75), [0.75,0.85), [0.85,0.95), and for each range compute the average number of comparisons (i.e., the number of positions you must check before you find an empty one).
- Compare the two hashing techniques in terms of time and space compplexity.

To turn in the project write

`turnin -c cs251_10:30 -p project2` filenames
`turnin -c cs251_1:30 -p project2` filenames

whichever applies.