

## Assignment #8 - Word Count w/Hash Tables

**Date assigned:** Tuesday, November 24, 2009

**Date Due:** Tuesday, December 8, 2009

**Points: 75**

Remember, there is no late grace period for this last assignment!! Also, I must receive your printout by **9:40am, Tuesday, December 8, 2009**.

You are to write a C program that produces a total count for the number of times each word is found in a plain text file by using a hash table to store each word. Word breaks are identified by non-alpha characters. Further, you are to convert all text to lowercase before hashing the text into the hash table. Your output is to be in sorted order as described below with each word listed and a count of the number of times the word appeared in the text document under properly labeled column headings. Finally, print the summary statistics described below.

Specifics:

1. You are to use the Mid Square hash method for hashing keys into your hash table. Compute the key value by summing the ASCII values for each character. (No, this is not a good way to compute a key value. As we discussed in class, a better way is to give more weight to the ending characters.) Take that value and square it. Use bits 8 through 17 of the 32-bit key value for hashing into the table.
2. Implement chaining using dynamic memory allocation as your collision handling technique.
3. You must decide how you are going to sort the words efficiently. Be careful. An inefficient method will lose several points. Further, your program is to print out the words in one of two ways:
  - a. Sorted alphabetically (increasing) based on the string (not key) values. The command prompt is **wordcount file -a**
  - b. Sorted from the highest count total to the smallest count total. The command prompt is **wordcount file -c**. Words with the same count total are to be displayed alphabetically increasing.
4. Output is to be properly labeled and displayed on the display screen.
5. If key K2 hashes to the same location as key K1 and key K1 is already in the hash table, place key K2 at the front of the chain. Implement this strategy for all collisions.
6. As summary statistics, your program must output:
  - a. The total number of unique words hashed into the hash table.
  - b. The total number of words in the file.
  - c. The number of empty slots in the hash table.
  - d. The average number of probes to find an arbitrary key K.
  - e. The length of the longest chain.

I will place a copy of the text files to be used in testing this program on the Web a couple of days before the assignment is due. If you have any questions or

anything is unclear, please let me know.

Here is an example of what your output is to look like for:

```
wordcount filename -a
```

```
1234567890123456789012345678901234567890 (do not print this line)
```

```
*****
```

```
*                               WORD FREQUENCY RESULTS                               *
```

```
*****
```

WORD	COUNT
----	-----
IS	10
NOW	5
THE	23
TIME	1

and so on

```
UNIQUE WORDS:    XXXXX  
TOTAL WORDS:    XXXXX  
EMPTY SLOTS:    XXXXX  
AVERAGE PROBES: XX.XX  
LONGEST CHAIN:  XXXXX
```

## **NOTES**

1. On the day the assignment is due, turn in a hard copy of your .h files followed by your .c files.
2. Submit your tarball as 08yourlastname.tar.gz. When extracted, I will change into the WordCount directory, typing **make clean**, then **make**, then **./wordcount file -a** OR **./wordcount file -c** after remaking all other modules.
3. Reuse as much of your other code as possible.
4. Extra Credit (5 pts) - If you use the qsort api in C as your only sorting function, you can receive up to 5 extra credit points for this assignment. Get the program to work correctly before trying this option!!! Also, please do not discuss amongst yourselves how to use the api as it is extra credit.