Programming Assignment 6

Hash Table Implementation of a Concordance

Due March 9, 2012 by 11:59 PM

In this assignment there will be a programming portion and a written portion.

PROGRAMMING

For this assignment, you will complete the hash-map with chaining as specified by the provided header file. Chapter 12 and worksheet 38 will be good resources for completing this assignment. After completing the hash-map implementation you will write a concordance program in your main.c. A concordance counts the number of occurrences of each word in a document. For this assignment, we will work with text files.

There are three files to work with for the programming portion:

- main.c is where you will write the concordance code.
- hashMap.h the header file which defines structs and public functions for your hash table and, explains the purpose of each function. Notice that there is a variable HASHING_FUNCTION which determines which hashing function should be used in hashMap.c Your code should check this value to determine which hashing function to use. HASHING_FUNCTION==1 means use stringHash1 and HASHING_FUNCTION==2 means use stringHash2.
- hashMap.c a mostly empty implementation file which you will fill in with your code.
- Makefile Remember to rename this from Makefile.txt to makefile when you 'save as'

You will be making changes to main.c and hashMap.c. Do not make changes to hashMap.h

There are three functions provided to you:

- `getWord(FILE*)` will parse out the next word from the file for you. Read the description of the function in the comments inside main.c for more information.
- `stringHash1(char*)` is the first function you will use for converting a key into an integer hash index. This function is located in the provided hashMap.c.
- `stringHash2(char*)` is the second function for computing an integer hash index, part of your job will be to explain why stringHash2 is better then stringHash1. This function is located in the provided hashMap.c.

There is some code in the main function for giving you timing information as well. You should not need to change anything in order to use it. The purpose of the timing information will be made clear in the questions section.

The worksheets and the function explanations should provide the information needed to complete this assignment, except for some more details on what a concordance is.

Concordance

The job of the concordance is to count how many times each word occurs in a document. You will implement a concordance using a hash table implementation of the Map interface. In this implementation, the hash table will store hashLinks which consist of a key, value, and pointer to the next link in the chain. The keys are the words and the values are the number of occurrences of each word.

CS261 Home
Schedule
Assignments
Resources
Policies
Grades

admin
edit SideBar

You are provided with a function to retrieve words from a FILE pointer. It is your job to open this file (fopen()) and close this file (fclose()) but the reading of the file will be handled for you inside of getWord. For help with fopen() and fclose() see the slides posted on the schedule for the recitation day.

Your concordance will run a loop until the end of the file is reached. In the loop, you will:

1. Read in a word with getWord().
2. If the word is already in your hash table then increment it's number of occurrences.
3. If the word is not in your hash table then insert it with an occurrence count of 1.

After processing the text file into your concordance you will print all the words in your hash table. Please print the words in the following form with only one word on each line:

for the input file of: `It was the best of times, It was the worst of times.`

```
best: 1
It: 2
was: 2
the: 2
of: 2
worst: 1
times: 2
```

You may choose any order in which to print the words.

Challenge

There are a lot of uses for a hashMap, and one of them is for implementing a spell-checker. All you need to get started is a dictionary.

Dictionary

spellcheck.c

Inside spellcheck.c you will find some code to get you started, but it should look very much like main.c

Written Questions

Your written questions will be handed in electronically, preferably as comments on the TEACH turn-in page, just cut and paste from your preferred editor.

1. Give an example of two words that would hash to the same value using `stringHash1()` but would not using `stringHash2()`.
2. Why does the above make `stringHash2()` superior to `stringHash1()`?
3. When you run your program on the same input file but one run using `stringHash1()` and on the other run using `stringHash2()`. Is it possible for your `size()` function to return different values?
4. When you run your program on the same input file using `stringHash1()` on one run and using `stringHash2()` on another, is it possible for your `tableLoad()` function to return different values?
5. When you run your program on the same input file with one run using `stringHash1()` and the other run using `stringHash2()`, is it possible for your `emptyBuckets()` function to return different values?
6. Is there any difference in the number of 'empty buckets' when you change the table size from an even number, like 1000 to a prime like 997 ?
7. Using the timing code provided to you. Run you code on different size hash tables. How does affecting the hash table size change your performance?

What to submit

1. main.c
2. hashMap.c
3. spellcheck.c
4. The answers to the written questions as either a teach comment or .pdf