CS 260 - Homework 04

Introduction

In the past two assignments, you have been provided templates to work with. This time, you get to design the Data Structure on your own. You must meet a few requirements for your **Abstract Data Type**, but you can decide how best to implement the solution. You are not given *enough* functions to build the entire project. You will need to design more on your own. You are just required to *at least* implement the ones listed below for full credit.

Open Hash Tables are very useful when you want to find unique items in a dataset without sorting. You could sort the whole set and then remove duplicates, but if you never need the values sorted this is a waste of effort. An Open Table can give us a quick way to determine if a value is a duplicate or not without sorting the data. This is great for collecting like objects together.

You will create a program what finds all the unique words in a text file. It will print out the total number of unique words and the total number of words. You are determining the count of words ignoring duplicates.

The program will ask for two values from the user.

- The size of the Open Hash Table (an integer)
- The name of the file to read in (a string)

You must solve this problem using an Open Hash Table.

Note: You may reuse any code you wrote (or were provided) in previous assignments.

Simplifying Assumptions

In the real world, users make all kids of mistakes. You may assume the following things will be true. Focus on the data structures, not weird input errors.

- The size given will never be 0 or negative.
- The file will always exist.
- No word or the file name will be longer than 254 characters.

Open Hash Table

Your Open Hash Table **must** have the following interface. You are free to add any additional methods, but you must *at least* implement these.

```
/**
    Create a new Hash table based around an array
    of size elements.
    @param size is the size the array the table is built with
    @return a pointer to the struct
```

```
*/
OpenHash* newOpenHash(int size);

/**
    The hash function for strings.
    @param word is a null terminated character array
    @param n is the size of the array
    @return the hashed index for the string
*/
int hash(char* word, int n);

/**
    Ask if string is in the table
    @param word is a null terminated character array
    @param table is the Hash Table to search
    @return true if found and false if not
*/
bool member(char* word, OpenHash* table);

/**
    Insert a string into the table. Ignore duplicates.
    @param word is a null terminated character array
    @param table is the Hash Table to search
*/
void insert(char* word, OpenHash* table);
```

Everyone needs to use the same Hash algorithm. Otherwise, the assignment could not be autograded. Here is a hash for strings.

```
/**
    Hash a string to a position in a hash table.
    @param word is the string to hash
    @param n is the size of the target hash table
    @return an index into the hash table as an integer (0 to (n-1))
*/
int hash(char* word, int n){
    int total = 0;
    for(int i=0; word[i]!=0; i++){
        char c = word[i];
        total = total+(int) c;
        total = total*101;
        total = total%n;
    }
    return total;
}
```

Word Count

Given a text file as input, we will count the number of unique words that appear. We want to count a word the first time we see it, then we never count the same word again. An open hash table is a great Data Structure to solve this problem.

The basic idea is described below.

```
For each word in the file do
    If the word is not in Hash Table then
        Count Unique Word
        Add to Hash Table
    End If
    Count Total Words
End For
Print Total Words in File
Print Unique Words in File
Print Hash Table Stats
```

We need to define what a "word" is. There may be many special characters and numbers in a file. Here are the rules for defining a word:

- Words are only broken by Space, Newline, Return, or End Of File
- Case Doesn't matter, convert all words to lowercase
- Special Characters and numbers get ignored

Imagine we have the following sentence: "They aren't going to buy 17 cats."

We would call the **words** in this file:

1. they (*T became lower case*)
2. arent (*single quote is ignored*)
3. going
4. to
5. buy
6. cats (*period is ignored*)

Notice that 17 was completely ignored because it contains no letters, only numbers.

Your program will take two inputs from the user. The first is the size of the array as an integer. This tells you how may spaces to make the array forming the foundation of your Open Hash Table. The second is the file to read.

You may assume the user **will never** call the program with bad inputs.

We want to see how well the Hash Table did. Once the program is over, print out some statistics about the table. For each row, print out how many values were placed in that row. Then print out the average number of values per row.

At the end of this document, all unit tests for the autograder will be provided.

Example:

```
Enter Size of Hash Table:
10
Enter Name of File:
examples/001.txt
Total Words: 18
Unique Words: 6
Hash Size: 10
Row 0 contains 0 values.
Row 1 contains 0 values.
Row 2 contains 2 values.
Row 3 contains 0 values.
Row 4 contains 0 values.
Row 5 contains 0 values.
Row 6 contains 1 values.
Row 7 contains 2 values.
Row 8 contains 1 values.
Row 9 contains 0 values.
Average Length: 0.60
```

Rubric

Component	Points
Doxgygen Comments	4
Style Guidelines	4
newOpenHash	8
hash	5
member	15
insert	15
Unit Test 00	7
Unit Test 01	7
Unit Test 02	7
Unit Test 03	7
Unit Test 04	7
Unit Test 05	7
Unit Test 06	7

Extended Examples

Below is all the tests the autograder will do.

Unit Test 00

```
Enter Size of Hash Table:
10
Enter Name of File:
examples/001.txt
Total Words: 18
Unique Words: 6
Hash Size: 10
Row 0 contains 0 values.
Row 1 contains 0 values.
Row 2 contains 2 values.
Row 3 contains 0 values.
Row 4 contains 0 values.
Row 5 contains 0 values.
Row 6 contains 1 values.
Row 7 contains 2 values.
Row 8 contains 1 values.
Row 9 contains 0 values.
Average Length: 0.60
```

Unit Test 01

```
Enter Size of Hash Table:
10
Enter Name of File:
examples/345-0.txt
Total Words: 163425
Unique Words: 10713
Hash Size: 10
Row 0 contains 1066 values.
Row 1 contains 1080 values.
Row 2 contains 1084 values.
Row 3 contains 1062 values.
Row 4 contains 1094 values.
Row 5 contains 1058 values.
Row 6 contains 1060 values.
Row 7 contains 1132 values.
Row 8 contains 1023 values.
Row 9 contains 1054 values.
Average Length: 1071.30
```

Unit Test 02

```
Enter Size of Hash Table:
50
Enter Name of File:
```

```
examples/345-0.txt
Total Words: 163425
Unique Words: 10713
Hash Size: 50
Row 0 contains 210 values.
Row 1 contains 228 values.
Row 2 contains 195 values.
Row 3 contains 228 values.
Row 4 contains 221 values.
Row 5 contains 231 values.
Row 6 contains 215 values.
Row 7 contains 209 values.
Row 8 contains 169 values.
Row 9 contains 202 values.
Row 10 contains 215 values.
Row 11 contains 204 values.
Row 12 contains 202 values.
Row 13 contains 213 values.
Row 14 contains 213 values.
Row 15 contains 190 values.
Row 16 contains 210 values.
Row 17 contains 241 values.
Row 18 contains 205 values.
Row 19 contains 209 values.
Row 20 contains 205 values.
Row 21 contains 221 values.
Row 22 contains 223 values.
Row 23 contains 185 values.
Row 24 contains 198 values.
Row 25 contains 204 values.
Row 26 contains 202 values.
Row 27 contains 231 values.
Row 28 contains 200 values.
Row 29 contains 204 values.
Row 30 contains 229 values.
Row 31 contains 219 values.
Row 32 contains 245 values.
Row 33 contains 217 values.
Row 34 contains 236 values.
Row 35 contains 220 values.
Row 36 contains 231 values.
Row 37 contains 222 values.
Row 38 contains 232 values.
Row 39 contains 211 values.
Row 40 contains 207 values.
Row 41 contains 208 values.
```

```
Row 42 contains 219 values.
Row 43 contains 219 values.
Row 44 contains 226 values.
Row 45 contains 213 values.
Row 46 contains 202 values.
Row 47 contains 229 values.
Row 48 contains 217 values.
Row 49 contains 228 values.
Average Length: 214.26
```

Unit Test 03

```
bin/wordCount 10 examples/1080-0.txt
Total Words: 6468
Unique Words: 1578
Hash Size: 10
Row 0 contains 162 values.
Row 1 contains 159 values.
Row 2 contains 158 values.
Row 3 contains 157 values.
Row 4 contains 160 values.
Row 5 contains 163 values.
Row 6 contains 150 values.
Row 7 contains 174 values.
Row 8 contains 153 values.
Row 9 contains 142 values.
Average Length: 157.80
```

Unit Test 04

```
bin/wordCount 50 examples/1080-0.txt
Total Words: 6468
Unique Words: 1578
Hash Size: 50
Row 0 contains 30 values.
Row 1 contains 33 values.
Row 2 contains 32 values.
Row 3 contains 27 values.
Row 4 contains 32 values.
Row 5 contains 30 values.
Row 6 contains 29 values.
Row 7 contains 34 values.
Row 8 contains 21 values.
Row 9 contains 31 values.
Row 10 contains 34 values.
Row 11 contains 29 values.
```

```
Row 12 contains 29 values.
Row 13 contains 37 values.
Row 14 contains 33 values.
Row 15 contains 42 values.
Row 16 contains 27 values.
Row 17 contains 35 values.
Row 18 contains 37 values.
Row 19 contains 27 values.
Row 20 contains 29 values.
Row 21 contains 32 values.
Row 22 contains 32 values.
Row 23 contains 20 values.
Row 24 contains 32 values.
Row 25 contains 30 values.
Row 26 contains 25 values.
Row 27 contains 38 values.
Row 28 contains 28 values.
Row 29 contains 30 values.
Row 30 contains 34 values.
Row 31 contains 38 values.
Row 32 contains 35 values.
Row 33 contains 36 values.
Row 34 contains 26 values.
Row 35 contains 34 values.
Row 36 contains 36 values.
Row 37 contains 39 values.
Row 38 contains 33 values.
Row 39 contains 30 values.
Row 40 contains 35 values.
Row 41 contains 27 values.
Row 42 contains 30 values.
Row 43 contains 37 values.
Row 44 contains 37 values.
Row 45 contains 27 values.
Row 46 contains 33 values.
Row 47 contains 28 values.
Row 48 contains 34 values.
Row 49 contains 24 values.
Average Length: 31.56
```

Unit Test 05

```
bin/wordCount 10 examples/pg2265.txt
Total Words: 32154
Unique Words: 5302
Hash Size: 10
```

```
Row 0 contains 551 values.
Row 1 contains 486 values.
Row 2 contains 525 values.
Row 3 contains 540 values.
Row 4 contains 476 values.
Row 5 contains 551 values.
Row 6 contains 541 values.
Row 7 contains 588 values.
Row 8 contains 520 values.
Row 9 contains 524 values.
Average Length: 530.20
```

Unit Test 06

```
bin/wordCount 50 examples/pg2265.txt
Total Words: 32154
Unique Words: 5302
Hash Size: 50
Row 0 contains 107 values.
Row 1 contains 110 values.
Row 2 contains 107 values.
Row 3 contains 108 values.
Row 4 contains 95 values.
Row 5 contains 120 values.
Row 6 contains 110 values.
Row 7 contains 113 values.
Row 8 contains 87 values.
Row 9 contains 97 values.
Row 10 contains 113 values.
Row 11 contains 84 values.
Row 12 contains 89 values.
Row 13 contains 117 values.
Row 14 contains 99 values.
Row 15 contains 91 values.
Row 16 contains 91 values.
Row 17 contains 107 values.
Row 18 contains 107 values.
Row 19 contains 97 values.
Row 20 contains 105 values.
Row 21 contains 77 values.
Row 22 contains 98 values.
Row 23 contains 95 values.
Row 24 contains 87 values.
Row 25 contains 115 values.
Row 26 contains 103 values.
Row 27 contains 118 values.
```

```
Row 28 contains 108 values.
Row 29 contains 109 values.
Row 30 contains 120 values.
Row 31 contains 102 values.
Row 32 contains 133 values.
Row 33 contains 113 values.
Row 34 contains 100 values.
Row 35 contains 130 values.
Row 36 contains 133 values.
Row 37 contains 138 values.
Row 38 contains 106 values.
Row 39 contains 109 values.
Row 40 contains 106 values.
Row 41 contains 113 values.
Row 42 contains 98 values.
Row 43 contains 107 values.
Row 44 contains 95 values.
Row 45 contains 95 values.
Row 46 contains 104 values.
Row 47 contains 112 values.
Row 48 contains 112 values.
Row 49 contains 112 values.
Average Length: 106.04
```