CS 340 Project III: Hashing Efficiency Based on Table Size

Description: For this project you are being supplied a file, LittleWomen.txt, which is the full text of the famous novel "Little Women." To make things easier, all punctuation has been removed, leaving only words. Some words, like "you'd", are rendered as "youd" because of the lack of punctuation.

For this project, you must write a program to create a hash table of each word used in this novel, along with the number times the word is used and the number of steps required to find or insert the word in the hash table.

Specifications:
You have the option of doing this project in C++, C# or Java. The instructions below are language-neutral. You should adjust them to fit the language of your choice.

The Java String class implements a hash function. For any String, the command myString.hashCode() hashes the string and returns a value. The hash may be negative, so for this assignment it's better to use the absolute value of the hash. Convert all letters to lowercase so that different capitalizations hash to the same value.

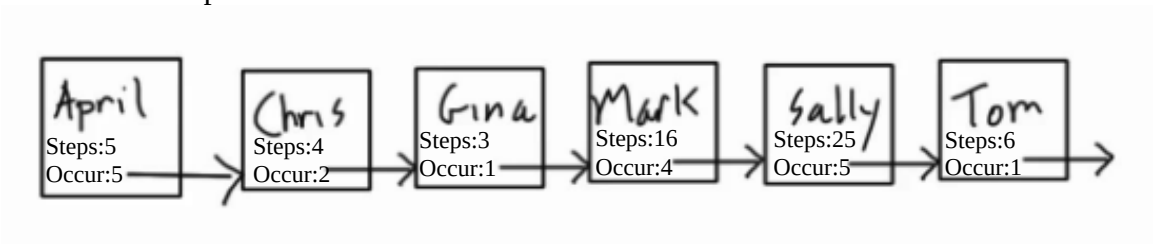
Here is C++ code that mimics Java's hash function. You may use this code or rewrite it in the language of your choice:

```
int HashCode (const std::string &str) {
   int h = 0;
   for (size_t i = 0; i < str.size(); ++i)
      h = h * 31 + static_cast<int>(str[i]);
   return h;
}
```

Your program should create hash tables of various sizes: 500, 1000, 2000, 5000 and 10000 slots. The hash table should use an array (or vector, or arraylist) of linked lists to keep track of the words that hash to that slot. It should read each word in the novel, then search for or insert the word in the hash table based on the hash function: absolute_ value(myString.hashCode()%tablesize).

The program should use linked lists to store collisions. The linked list should consist of nodes that record the key (the word), the number of steps it took to find the key in the list, and the number of occurrences of the key. You do not need to write your own linked list, but you need to be able to keep track of how many items you have searched through on the list. If you get to the end of the linked list without finding a word, you should add the word to the end of the linked list. On each pass your program should keep track of the number of occurrences of the current word, and the number of steps it took to find or insert the word. Number of steps is based on the position of the word in the linked list. If the word is first, it took 1 step to find. If the word is second, it took 2 steps to find, etc.

As an example, in the figure, if we are looking for the word “Chris,” your program should record that it took 2 steps to find the word, and increment occurrences by 1. In the example, the word has been found twice, and it is 2 steps in. The number of steps is 4, because 2x2 = 4. If we are looking for the word “Ami,” your program should add it to the end of the list and record 7 steps.



The program should print a file that contains each word, the number of occurrences of that word, and the number of steps it took to find the word. Make the file a CSV so that you can read it in Excel. Your CSV opened in Excel might look something like this:

	A	B	C	D
1	my	842	842	
2	escort	10	20	
3	several	51	153	
4	laid	46	184	
5	short	41	205	
6	month	20	120	
7	companion	3	21	
8	mending	7	56	
9	confessed	9	81	

Column B shows the number of occurrences of a word, and column C shows total number of steps to reach that word. From the example above, “my” is the first word in its chain. The word “several” has 51 occurrences and is the third word in its chain (because 51x3=153). Use Excel to calculate statistics such as the total number of words, the total number of occurrences, total steps, and average steps to find a word. Make a chart that plots the average number of steps against the size of the hash table.

In this project, you must use good object-orient design principles. You should create a class or struct that holds a word, its number of occurrences, and the total number of steps to record those occurrences. You should create a second class that has the hash table as an instance variable, and contains a method to add items to the hash table. Your main() method should be in a third separate class. Note that the programming for this project is very simple and isn’t worth many points. The project is more about figuring out hash tables and how well they work.

Your submitted project must contain a report that discusses the following:

1) Your report should begin with the following data (basic info):

- Give the total number of steps it took to find or insert a word.
- Give the average (or "expected") number of steps it took to find or insert a word.
- Discuss how many steps it would take to access all the words in the best possible circumstances versus how many it actually took?
- Discuss whether the data are as expected (in the statistical sense).
- Are average steps higher or lower than expected (in the statistical sense)? Why?
- You have added new words to the end of the LinkedList. Does this affect the expected behavior of the algorithm? Is there a way to determine which words were first in their linked lists?

2) In addition to the data above, a large part of the grade will be based on discovering something new concerning hashing. You must run an original experiment. For example,

- How do the numbers from part 1 change if words are added to the beginning of the linked list instead of to the end?
- How does the number of steps change if the words are held in a binary search tree or red-black tree instead of a linked list? How many steps does theory predict as opposed to how many are actually required?
- What happens if you change the hash function (write your own) and why?
- If you choose another novel, are the most frequent words substantially the same?
- Make up your own experiment.

3) **One plot must be included in your report:** Plot the average number of steps to find/insert a word against the size of the hash table.

4) Summarize: Given the data, how well does hashing work in real life situations? What influences the success/failure of hashing?

Grading will be based on the following rubric:

Programming	0 pts Program doesn't compile	5 pts Program has major errors	10 pts Program has minor errors	15 pts Program correct and professional	
Basic Information	0 pts No basic info	5 pts Basic info appears to be incorrect	10 pts Basic info correct	15 pts Basic info correct and presented well	
Originality of Experiment	0 pts No experiment	5 pts Unsuccessful experiment	10 pts Successful but unoriginal experiment	20 pts Experiment successful, interesting and original	25 pts Original experiment described well in the report.
Plot	0 pts No plot	5 pts Unsuccessful plot	10 pts Plot correct		
Summary	0 pts No summary	5 pts Summary has errors	10 pts Summary is correct		