1.
Timothy Ha
junkwan
1367917
junkwan@uw.edu
HW4
05.13.15

2. a) I was testing all throughout my coding. For the String Comparator I just tested it using any sort of random case I could think of and tested it against the String compareTo method. I didn't know whether my getCountsArray method was functioning correctly until I finished the HashTable classes first. I made two separate tester classes that I kept changing for my two implementations of the hashtables, separate chaining and open addressing. For the separate chaining I had to do a lot of testing to see whether my "chaining" was actually working or not. In addition to that, it was a little bit more difficult because I not only had to find an index in the hashtable, but then might have had to traverse through a linked list making it more two dimensional than the open addressing hashtable.
   b) For both hashtable implementations I had to test vigorously for adding two elements with the same value. I kept getting errors and I realized that I was incrementing size when I shouldn't have been because I was only adding one unique element instead of two, this really threw me off and so I continued to test for other various cases and to see when I should and should not increment things such as size. Also, for the simple iterator I had to look at the beginning and ends of the counter because I could run into indexoutofbound exceptions.

3.

| time (ns) | Hamlet | The New Atlantis |
|---|---|---|
| Separate Chaining | 262830000 | 216380000 |
| Open Addressing | 289641000 | 233375000 |

These were rounded averages of five trials for timing of Hamlet and The New Atlantis for both implementations of the hashtable. I just timed in the countWords method because that is when the elements are added.

c) I guess I initially thought OA would be faster for some reason, but in the end SC was faster. This makes sense because OA might have to go through multiple probing times AND it has to rehash/resize more often because the load factor for OA was 0.5 while the load factor for SC was 1.5. According to this data alone, and from the tests I have done, I say that separate chaining is better in this context.

5. Hamlet vs Romeo and Juliet: 2.6470538264355856E-4
   The essays vs the new atlantis: 3.922582862365105E-4

6. Not applicable
7. Not applicable
8. a) The hardest part of the assignment was definitely fully understanding what the spec was asking me to do. I found out that as I was going through this assignment, I kept missing many things, such as not rehashing correctly, I was just copying, when I resized the hashes. It was also difficult to take existing code, WordCount.java, and not just add on to it, but understanding the rest of the code so I knew what exactly I was supposed to code for the iterator. Also, it helped to debug because I couldn't get the java files to run through the terminal so I had to manually adjust some parts of WordCount.java and then change them back.
   b) I think I had a lot of confusion and trouble dealing with the terminal and trying to run "java WordCount -o hamlet.txt" for an example. Even after looking it up online and changing directories and compiling I just could not get it to work. I wish there were either clearer instructions for that or it was not even included at all. I did find a way around it by just manually changing some values in WordCount.java to make sure my hash tables were functioning correctly.