CSE 373 Homework 4 Write up

1. **Who is in your group (Give name, UW NetID & student number of each person)?**
   Morgan Evans, mnevans, 1124703
   Joshua Malters, maltersj, 1336144

2. **a) How did you design your tests & what properties did you test?**

   We designed our tests based on our implemented methods in the HashTable_OA and HashTable_SC programs, and tested the size (number of String elements) of the hash table, the corresponding getCount() of the String passed in, which also tests incCount(), and that the iterator iterates through the correct number of elements being inserted.

   b) What boundary cases did you consider?

   We considered the table size for the hash table methods, as this determines the load factor and indicates whether or not we need to rehash the table. We also considered whether or not a table is filled or there is a collision, and we need to begin chaining elements in the separate chaining table or probe the open addressing table.

3. **Conduct an experiment to determine which DataCounter implementation (HashTable_SC, HashTable_OP) is better for large input texts.**
   a) Describe your experimental setup:
   1) Inputs used

   The inputs we used are hamlet.txt and the-new-atlantis.txt.

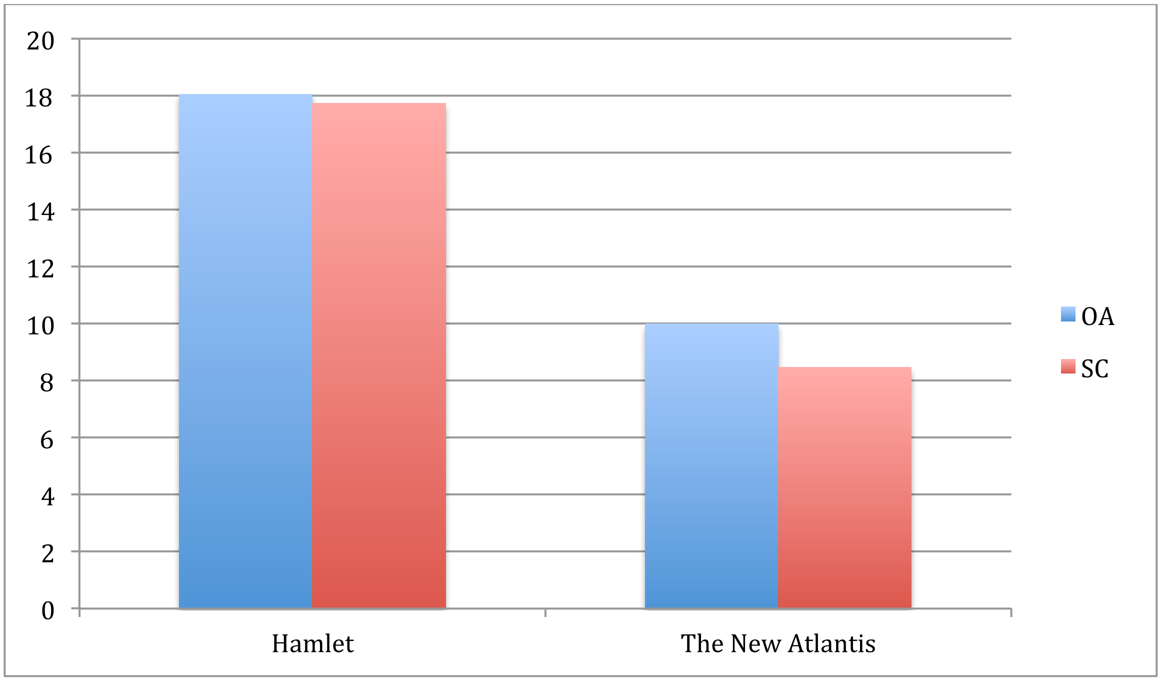
   2) How you collected timing information

   We collected timing information by calling a method, that takes in String file and DataCounter parameters, on both Hamlet and The New Atlantis, for both our OA and SC hash table programs. The method reads in a file and uses a while loop to loop over each word in the selected file. We begin the time at the insertion of the first word into the hash table and end it at the insertion of the last word, then sum up the times for a total elapsed time. We then print the total insertion time for both txt files for both hash tables.

   3) Any details that would be needed to replicate your experiments

   b) Experimental Results (Place your graphs and tables of results here).

	Hamlet	**The New Atlantis**
Separate Chaining	17.734 milliseconds	8.467 milliseconds
Open Addressing	18.06 milliseconds	10.0 milliseconds



c) Interpretation of Experimental Results
1) What did you expect about the results and why?

We expected the time to be longer to insert the elements into the open addressing tables because of the time taken to quadratic probe the table, as opposed to creating a linked list.

2) Did your results agree with your expectations? Why or Why not?

Yes, our results did agree with our expectations, as the time taken to insert the words into the open addressing table is far more than that of the separate chaining table.

3) According to your experiment, which Hashtable implementation, separate chaining or open addressing, is better?

According to our experiment, separate chaining is better as a Hashtable implementation.

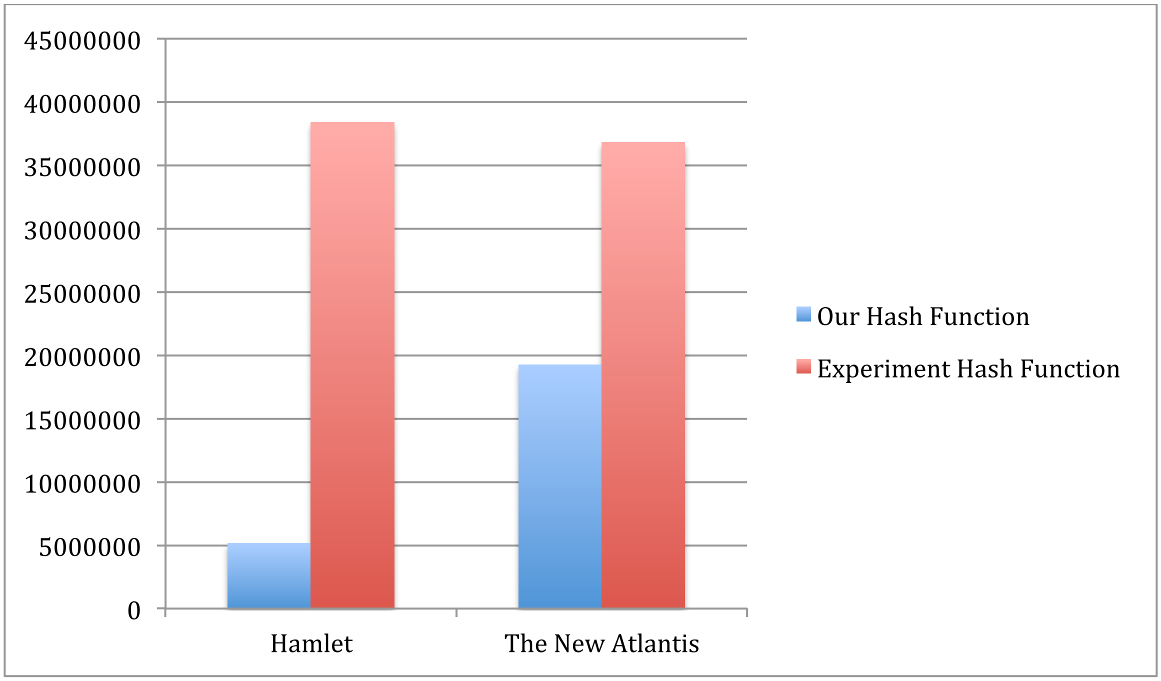
4. Conduct experiments to determine if changing the hash function affects the runtime of your HashTable.
a) Brief description of your hash functions

Our hash function loops through all of the characters in a string, keeping a running sum of the ASCII values of each character. It then multiplies that running sum by the length of the string and returns this sum.

Our experimental hash function loops through all of the characters in a string, keeping a running sum of the ASCII values of each character multiplied by 37 (a prime number).

b) Experimental Results (Place your graphs and tables of results here).
Experiment with at least 2 hash functions (2 Hashing functions = 2 experiments depending on how you measured the runtime)
Don't forget to give each graph a title and label the axes.

	Hamlet	**The New Atlantis**
Our Hash Function	5144999 nanoseconds	19240296 nanoseconds
Experiment Hash Function	38406381 nanoseconds	36813303 nanoseconds



c) Interpretation (Your expectations and why? Did it match your results? If not, why?)

We expected our hash function to have a faster run time because there are not as many calculations for it to make, as our's only multiplies by the word length at the end of the function, and the experiment function multiplies the word length by a constant number inside of the for loop. Yes, our results matched our expectation.

5. **Using Correlator, does your experimentation suggest that Bacon wrote Shakespeare's plays? Show at least one (you can experiment with more texts if you want) correlation value for each of:**
   a) Shakespeare's work compared to Shakespeare's work

   Hamlet versus Romeo and Juliet → 2.6204557854975593E-4

   b) Bacon's work compared to Bacon's work

   The New Atlantis versus The Advancement of Learning → 3.80603669585748E-4

   c) Shakespeare's work compared to Bacon's work

   Hamlet versus The New Atlantis → 5.657273669233966E-4

   According to the results of your experiments, did Bacon write Shakespeare's plays?

   NO

6. **Include a description of how your project goes "above and beyond" the basic requirements (if it does).**
   Our project does not go "above and beyond" the basic, functioning requirements.

7. <u>**If you worked with a partner**</u>**:**
   a) Describe the process you used for developing and testing your code. If you divided it, describe that. If you did everything together, describe the actual process used (eg. how long you talked about what, what order you wrote and tested, and how long it took).

   We worked both together and separately for some programs. We began by coding "rough drafts" of each method in the separate chaining program, and then did the same for the open addressing one. This included some talking through each method, but was more to get our ideas down. We then wrote the testing code to test each hash table method and tested it to find that our getCount(), incCount() and getIterator() methods all had bugs. From here, we drew out the processes of the methods to visualize how the values were being inserted in to the tables. This allowed us to identify and fix several errors (such as a missing index++ in the SC iterator). The remaining errors we were able to fix via debugging and following the code as it hashed values. A big one we discovered was that we were receiving a collision at the value of 15 for separate chaining, and our count stopped incrementing. We fixed this with a simple "node = node.next" line. Overall, this entire process took us about 8 hours. We then ran the StringHasher and StringComparator programs, and those both had errors as well. Josh quickly found that our for loop statement in StringHasher had one too many variables (we incorporated the table size while this was not actually necessary), and StringComparator threw a null pointer exception which we found that in the getCountsArray() under WordCount, we were setting the array to the incorrect size. The timing code took only about **MINUTES** to write as we have had somewhat ample experience implementing this, but the more time consuming part is changing the value to be timed and extracting the results.

b) Describe each group member's contributions/responsibilities in the project.

We worked together on both Hash Table programs and testing code. Then Morgan did the timing code, and Josh did the StringHasher and StringComparator programs.

c) Describe at least one good thing and one bad thing about the process of working together.

One good thing was that we were able to bounce ideas off of each other and each person can provide his/her own point of view, and can notice errors that the original person did not notice (much like proof reading somebody's paper). One bad thing is that everybody has a different way of approaching a given problem and it can be difficult, even frustrating, to have to explain your train of thought to somebody else when it just "makes sense" to you.

8. a) Which parts of the project were most difficult?

The most difficult parts of the project were getting the hash tables to work correctly, as we ran into many errors here. We spent several hours tweaking potential errors and re-running our test code, and became tired and frustrated as we knew it was likely small one-line errors (which it was for the most part).

b) How could the project be better?

This project could be better with a more detailed spec on what each method in HashTable_OA and SC is doing and how to implement it. We found the notes left in the code to be somewhat vague and not too helpful.

Appendix

Place anything else that you want to add here.