Derek Aguirre

This assignment is an introduction to the concept of a parallel map reduce program. This program will search for a select number of words across various Shakespeare works. The concept of the map reduce approach is essentially breaking up the work between multiple threads to count the number of the words that will be searched. The following screenshots are the results of the execution times.

Threads	Results	Time
1 Thread	student@linux-4oae:~/Desktop/A2> python3 MapReduceA2.py Please enter thread amount (up to 8) >1 Executing the program with 1 threads Seconds elapsed for reading files: 0.011214733123779297 Thread 0 read a document for: 0.0033388137817382812 seconds Thread 0 read a document for: 0.03501248359680176 seconds Thread 0 read a document for: 0.009877920150756836 seconds Thread 0 read a document for: 0.0249326229095459 seconds Thread 0 read a document for: 0.029389619827270508 seconds Thread 0 read a document for: 0.03129315376281738 seconds Thread 0 read a document for: 0.024807453155517578 seconds Thread 0 read a document for: 0.022539377212524414 seconds Total elapsed time 2.92 seconds	**2.92** seconds
2 Threads	student@linux-4oae:~/Desktop/A2> python3 MapReduceA2.py Please enter thread amount (up to 8) >2 Executing the program with 2 threads Seconds elapsed for reading files: 0.011025428771972656 Thread 2 read a document for: 0.0032052993774414062 seconds Thread 1 read a document for: 0.009813547134399414 seconds Thread 2 read a document for: 0.033051490783691406 seconds Thread 1 read a document for: 0.024322986602783203 seconds Thread 2 read a document for: 0.028693199157714844 seconds Thread 1 read a document for: 0.03146696090698242 seconds Thread 2 read a document for: 0.024550199508666992 seconds Thread 1 read a document for: 0.022152423858642578 seconds Total elapsed time 1.58 seconds	**1.58** seconds
4 Threads	Executing the program with 4 threads Seconds elapsed for reading files: 0.012055635452270508 Thread 4 read a document for: 0.0033075809478759766 seconds Thread 4 read a document for: 0.009972810745239258 seconds Thread 1 read a document for: 0.02889537811279297 seconds Thread 2 read a document for: 0.041253089904785156 seconds Thread 3 read a document for: 0.031151533126831055 seconds Thread 4 read a document for: 0.031522512435913086 seconds Thread 2 read a document for: 0.02304363250732422 seconds Thread 1 read a document for: 0.02478170394897461 seconds Total elapsed time 1.04 seconds	**1.04** seconds
8 Threads	student@linux-4oae:~/Desktop/A2> python3 MapReduceA2.py Please enter thread amount (up to 8) >8 Executing the program with 8 threads Seconds elapsed for reading files: 0.012745141983032227 Thread 2 read a document for: 0.003371477127075195 3 seconds Thread 5 read a document for: 0.010085582733154297 seconds Thread 4 read a document for: 0.025837421417236328 seconds Thread 1 read a document for: 0.023221254348754883 seconds Thread 8 read a document for: 0.03560900688171387 seconds Thread 3 read a document for: 0.0302126407623291 seconds Thread 6 read a document for: 0.03217482566833496 seconds Thread 7 read a document for: 0.034509897232055664 seconds Total elapsed time 0.97 seconds	**0.97** seconds

Derek Aguirre

The following screenshot includes the results of all the words that the program was able to find.

```
Occurrences found of the following words:
hate : 257
love : 3070
death : 1016
night : 1402
sleep : 470
time : 1806
henry : 661
hamlet : 475
you : 23306
my : 14203
blood : 1009
poison : 139
macbeth : 288
king : 2835
heart : 1458
honest : 434
student@linux-4oae:~/Desktop/A2>
```

The main issue that I faced during this assignment had to do with the the threads and trying to figure out why the threads were effectively zero indexed. When I chose to run four threads it would only show threads running but thread zero would never be utilized outside of forcing the program to run on a single thread. I am not sure why this issue is happening; my best guess would be the way that I choose to pass thread numbers to the program to execute with. A way to work around that issue is just to input a number higher than the one you want to test for such as inputting nine threads when wanting to test eight, which will allow the program to all eight threads. The biggest challenge was trying to tune the regular expression to capture only what was needed but once figuring out how to do it the program reported higher numbers until it reached the results on the assignment's specifications.

The observations that I made are largely similar to the ones from the previous assignment with regards to the fact that there starts to be a diminishing return as more threads are invested into a single process. I suspect the reason for the first observation is due to the context switching between threads so the computation time may be slower when adding more threads due to the overhead from the operating system. The problem could be either context switching or waiting on thread synchronization in the background.

Here is my CPU info:

```
student@linux-4oae:~/Desktop/A2> ./cpuInfo.sh cpu
model name      : AMD Ryzen 5 5600X 6-Core Processor
      4      36     192
```