Derek Aguirre

This assignment is an extension of assignment 2 with more of a focus on a new way to parallelize our map reduce program. in this assignment MPI will be utilized to run the word count algorithm. The following screenshot includes the results of all the words that the program was able to find.

```
student@linux-4oae:~/Desktop/A2> python3 mpiCode.py

Occurrences found of the following words:
hate : 332
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 : 4545
heart : 1458
honest : 434
Total elapsed time:  0.8611849339940818
```

The main issue that I faced during this assignment was trying to conceptualize my implementation using MPI instead of PyMP. This problem led to many issues with converting implementation into MPI specification. also another issue that I run into was during the conversion of my word count algorithm. Originally all of my documents we're already read and stored and every threads that took a document made an effort to clean up the document independently before searching. The algorithm used a regular expression to help clean the documents but one major issue with the regular expression when converting it to MPI was that it nearly tripled the runtime. upon removing the regular expression the result did not change but the runtime drastically reduced.

An observation that I made pertains to the total runtime of the program. The word count algorithm running on PyMP on 8 threads (0.97 seconds) runs slightly slower than the implementation in MPI (0.86 seconds). I believe the reason for this is because MPI does a better job of managing threads even while sending messages between them.

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
```