LLD report

By: Ayan Das
Group Members: Ayan, Adeeb, Nasif

The project me and my team is working on is the word count theme, where we are using the python library regex to calculate the word frequency and determine the most repeated word. We are experimenting with threading to speed up this process, to determine what number of threads provides us with the fastest runtime. However, we need to keep in mind that more threads does not mean faster performance per say, beyond a certain number of threads, additional thread implementation simply leads to margin of diminishing returns. Therefore, out of 1-8 threads, our goal is to determine the optimal number of threads, as in the thread with the fastest runtime.

The following python code shows how we imported a text file from Gutenberg.org and calculated it's word frequency and determined the most repeated word:

```
# Import the Neccessary libraries we will need
from concurrent.futures import ProcessPoolExecutor, ThreadPoolExecutor
import numpy as np
import time
import matplotlib.pyplot as plt
import glob
from PIL import Image
import pandas as pd
import random
import string
import re
%matplotlib inline   #This is to install matplotlib package in case it
isn't already installed using Anaconda

#The following is how we import a text file and read it as a string
with open('/MemorialsOfOldDurham.txt', 'r') as file:
    data = file.read().replace('\n', '')

#view the text file
print(data)   #this will only print out a portion of the string, since it
is such a large string
type(data)       #we have successfully imported our text file and
converted it into a string
```

```
# We will be using wordDict to calculate word frequencies in python
from collections import defaultdict
''' Purpose of Default Dict:
A defaultdict works exactly like a normal dict, but it is initialized with
a function (“default factory”) that takes no arguments and provides the
default value for a nonexistent key.
 A defaultdict will never raise a KeyError. Any key that does not exist
gets the value returned by the default factory.  '''

#implement the word counting frequency algorithm
# function for the word frequency
counts = defaultdict(int)
def wordFrequency():
  for word in re.findall('\w+', data):
      counts[word] += 1

  #return the dictionary
  return dict(counts)

wordFrequency()  #returns the word frequency in the form of a dictionary

# Word with the most frequency
# The following is how we iterate through a dictionary


#implementation of the function with the highest word frequency
def HighestWordFrequency():
  max = 0
  key = ''
  for items in counts:
    if counts[items] > max:
      max = counts[items]
      key = items
  return str(items) + ' : ' + str(max)

#call the function
HighestWordFrequency()    #the most frequent word that appears is
newsletter, appears 6235 times within the book
```

The code shows how I implemented the search algorithm to calculate the word frequency in the text file and determine the most most frequently appeared word