Cheaters!
EE 312

100 points

Due: Tuesday 5/7/2019 at 10:00pm

This is a large project and it incorporates many of the concepts we have studied this semester.

Here is the original project description (.doc). We will not be implementing the entire project (e.g. a graphical interface). This is our vehicle for doing research on hashing and using other containers.

Here are the files that you will use for data.

- BigDocSet.zip - 1,354 docs -- ~1,615,193 words
- MedDocSet.zip - 75 docs -- ~91,273 words
- SmDocSet.zip - 25 docs -- ~33,946 words

Milestone I

You need to be able to process a set of documents in a directory and produce all possible n-word sequences. You should be able to change n relatively easily. Proof of this milestone consists of demonstrating you can print all n-word sequences to the console for a given n. You will not have to turn in the milestone separately, but it should be your first step on this assignment.

Write a program called "plagiarismCatcher" that will take command line parameters for the path from the executable program to the text files and n (the length of the word sequence).

e.g. prompt> ./plagiarismCatcher path/to/text/files 6

Here is a program that gives some help with getting the file names from a directory.

- You will need the c_str() function to open an input file using a C++ string variable.
- A queue is a useful container for creating length n chunks of words.
- Don't worry about capitalization, punctuation or strange characters.

Milestone II

The minimum here is step of identifying the suspicious cases on the small data set. A nice product would be a simple console application that accepts 3 command-line arguments. For example:

./plagiarismCatcher path/to/files 6 200

which would churn and then produce a list (in order) of all the pairs of files in path/to/docs that shared more than 200 6-word sequences in common.

Your final program should be able to produce meaningful output for at least the small set of documents (25 or so).

Lastly, with Milestone II you will submit a short document (the project README) about what your program does, how to use it, what works, what doesn't work and any other features, bugs I should know about when I'm looking at your code.

Hint: Hashing the six-word sequences and then looking for collisions is a good strategy for finding improper collaboration.

Notes:

- You will likely need a very large hash table (hundreds of thousands of entries?)
- Don't store the chunks in the hash table. Just the file names (or and index to the file name) that the chunk came from.
- Don't worry about small differences in the chunks (punctuation, etc.)
- You will likely need a 2-D array to store a count of the similarities between the files.

The output could look like this:

700: filenameA, filenameD
350: filenameA, filenameC
350: filenameC, filenameD
205: filenameB, filenameE
204: filenameA, filenameB

Notes:

- You may use one partner on this project.
- You must use GitHub to store your files (even if you are working by yourself).
- This program must compile and run on kamek.ece.utexas.edu.
- **You must name include a text file named READ_ME that describes how to use the makefile and the names of the files** .
  - **You can create whatever files and classes you need. Everything should compile with the makeifile you provide. The executable pogram should be called "plagiarismCatcher".**
- Be sure to follow the documentation standards for the course.

Turn in: Each partner should hand in a zipped file to Canvas named cheaters_xxxxxx.zip where xxxxxx is your UTEID.

Last Updated: 4/4/19