Cheaters! (10 points)

This is a big project and it incorporates many of the concepts we have studied this semester. This is also a vehicle for doing research on hashing and using other containers.

The goal of this assignment is to check for plagiarism in essays submitted by students. We will use a simple algorithm to do this. Here is a step-by-step approach that you will use:

1. Preprocess an essay (a text file) to convert upper case to lower case and remove all punctuation and special characters EXCEPT for the apostrophe ('). So the essay after pre-processing should be a string of words separated by spaces, all in all lower case.
2. Take a command line input variable p and parse the pre-processed essay to create a list of all unique p-word sequences in the essay
3. Compare the list of p-word sequences between two essays and count the number of common sequences
4. Repeat step 3 for every pair of essays. For a set of 25 essays, there will be 300 pairs (25 x 24 / 2)
5. Sort all the pairs in descending order of the number of common sequences. In other words, the top of the list should show the pair of essays that share the largest number of p-word sequences
6. Take a command line input variable m and print a sorted list (in descending order) of all the pairs of files that shared more than m p-word sequences in common. m is a threshold that serves as a "plagiarism suspicion index"

Example Output for $p = 6$ and $m = 200$:

```
1618: sra31.txt, jrf1109.txt
530: abf70402.txt, abf0704.txt
523: abf0704.txt, edo26.txt
384: catchmeifyoucan.txt, tyc12.txt
378: edo14.txt, bef1121.txt
311: catchmeifyoucan.txt, hal10.txt
285: ecu201.txt, catchmeifyoucan.txt
```

The files that you will use for data are zipped in sm_doc_set.zip, which contains 25 docs (~33,946 words).

OPTIONAL: We have also provided two other data sets with more files, med_doc_set.zip with 75 docs (~91,273 words) and big_doc_set.zip with 1,354 docs (~1,615,193 words), but processing these files is **NOT** a requirement for this assignment.

Tips/Recommendations

1. Write a program called “plagiarismCatcher” that will take command line parameters for the name of the directory that holds the text files, p (the length of the word sequence), and m, the “plagiarism suspicion index”.
   e.g., ***prompt>*** ./plagiarismCatcher sm_doc_set 6 200
   which would process the files and produce a sorted list (in descending order) of all the pairs of files in the sm_doc_set directory that has more than 200 6-word sequences in common.
2. The zip file has a program dir_help.cpp that should help with getting the file names from a directory
3. You will need to #include <fstream> to open the files and read them.
4. A queue is a useful container for storing length n sequences of words.
5. Hashing the six-word sequences and then looking for hash table collisions is a good strategy for finding matches
6. You should do some research to find good hashing functions for strings.
7. You are allowed to use the C++ STL hash function, but you will learn a lot by building your own hashing function
8. You are allowed to use containers from the C++ STL like set, pair, map, vector, etc.
9. You will need a very large hash table (tens of thousands of entries)
10. You don’t need to store the word sequences in the hash table. Just the file names (or an index to the file name) that the sequence came from.
11. You will need to create a 2-D array (25 x 25) to store a count of the similarities between the files
12. Your output will most likely not match the example output given above, because it will depend on your choice of the hashing function, and other implementation choices. We expect that your numbers should be within a +/- 200 range of the numbers above

---

Requirements:

- The requirement for this assignment is to process a small data set of documents (sm_doc_set.zip which contains 25 essays). The other two larger document sets are optional and not required for this assignment.
- The executable program should be called "plagiarismCatcher"
- This program must compile and run on kamek.ece.utexas.edu.
- You **must** include a text file named README that describes how to use the makefile and the names of the source files. The README file should describe what your program does, how to use it, what works, what doesn’t work and any other features, bugs we should know about when we’re looking at your code.
- You can create whatever files and classes you need. Everything should compile with the makefile you provide.
- Be sure to follow the documentation standards for the course.

Turn in: **A zipped file to Canvas named cheaters_xxxxxx.zip where xxxxxx is your UTEID.**

---

modified by Vivek Telang (vpt 7/12/23)

created by Roger Priebe (3/30/20 rlp)