Ivy Truong

2/4/2022

CSCI 3800

HW2

1. My view on plagiarism is that if the student uses code and does not site another person’s work, then it is copying someone else’s work and is cheating. To some extent, I think that it is necessary to use outside resources to help understand concepts from class, since it brings another perspective. I think it is also necessary if the student is really stuck on a certain part of the program and need a new perspective on how to solve a portion of the problem. Learning is all about trial and error, and when one gets stuck, be able to ask for help whether it is from the professor or an online resource.

In terms of reusing information, as long as the student sites where they received the information and can comment in the code what it does and how it correlates with the current problem being solved, then it is not plagiarism. I think that it becomes plagiarism if 90-100% of the code uses solely outside resources and does not have any critical thinking input from the student. Unless the assignment is to go through a tutorial, then all assignments that require problem solving and critical thinking from the student should have at least 75-85% of the student’s own work.

1. Taking files as input and displaying the top ‘n’ most used words in the file. I set n = 10 to showcase the examples.

Here is a description of what each command does in order of when they were executed:

1. Outputs the contents of the file
2. Takes the contents and deletes all characters that are not a-z, A-Z, 0-9, new lines, or spaces
3. Takes the contents and squeezes all spaces as new lines
4. Takes the contents and converts all uppercase letters to lowercase letters
5. Takes the contents and sorts the words
6. Takes the sorted words and gives a count of all unique words
7. Takes the output of the previous command (the counts/frequencies with the word) and sorts them by the count in reverse order.
8. Prints out the first 10 most frequent words

cat Lincoln.txt | tr -cd [a-zA-Z0-9"\n"[:space:]] | tr -s " " "\n" | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort -n -r | head -10

13 that

11 the

10 we

8 to

8 here

7 a

6 and

5 of

5 not

5 nation

cat King.txt | tr -cd [a-zA-Z0-9"\n"[:space:]] | tr -s " " "\n" | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort -n -r | head -10

102 the

99 of

60 to

41 and

38 a

33 we

32 be

26 will

24 that

23 is

cat Kennedy.txt | tr -cd [a-zA-Z0-9"\n"[:space:]] | tr -s " " "\n" | tr '[:upper:]' '[:lower:]' | sort | uniq -c | sort -n -r | head -10

32 the

24 of

20 to

19 we

15 and

12 a

10 our

9 that

7 not

7 in

1. a. When ‘sort file1 > file1’ was executed, the terminal outputted an error message. The command does not allow an existing file to be overwritten by modified data. The contents of the file after executing the command are still the same, without being modified.

b. Command to overwrite existing file: sort file1 -o file1

-Need to use the -o for overwriting if set -o nocobbler is defined. -o will sort first, then write back to original file. I also tried sort file1 >| file1, but it didn’t work because redirection has higher precedence.

Below are screenshots of what I did to show the command above works

1. copied file with the same contents of one of the example files (just in case I messed up later)

A picture containing graphical user interface

Description automatically generated

1. Did it with the first command ‘sort file1 > file1’ and here is the result

A picture containing text

Description automatically generated

1. Did it with modified command to overwrite contents ‘sort file1 -o file1 and here is the result’

Text

Description automatically generated

1. Extra Credit: Shell script file will be attached on canvas

Sample output below

Text

Description automatically generated