**Describe the state of your project, what works and what doesn’t.**

Built the program to read a large text file that stores the words in a list, and uses TreeMap and a HashMap to count the frequency of each word. The results print out the top 5 most frequent words that are longer than 6 characters. What I noticed is that short words or symbols still show up even if they're not filtered correctly, but for large files it didn’t really mess with the results much

**Describe how you tested your program**

I had downloaded the book from Project Gutenberg, called Woman-Through A Man’s Eyeglass by Malcolm C. Salaman. Had to clean the file by removing commas, periods, exclamation marks, and question marks. Ran the code on smaller text files first for the logic to work, then to the large file for actual timing. Running the program with the big file helped me check both correctness and performance under heavier load.

**Show your timing data and write a paragraph on the benefits/drawbacks of writing a custom sort routine**

HashMap was faster than TreeMap. HashMap is unpredictable because of how it’s ordered, when TreeMap keeps the keys in sort order.

**In a sentence or two, what did you learn?**

Learned how to store and process large amounts of text data with different map structures, and how performance can really change based on the data structure you pick.

**In a sentence or two, what did you like about this project? In a sentence or two, what did you find confusing or would like to see done differently regarding this project?**

Helped with thinking about performance and how to organize data better.

**In a sentence or two, if you had another hour or two, what would you like to add to the project or how would you do things differently?**

Could be nice to have a live word count tracker while it runs, like currently how in WORD Document it’s doing it as I am typing and typing and typing.