1. The static benchmark and benchmarkShift methods both generate the same result. How are they different? Specifically, answer the following about their differences:

(a) Which method reads the entire file before creating any WordGrams and which creates WordGrams while reading the file?

The benchmark method reads the entire file into a list before creating any WordGrams, while the benchmarkShift method creates WordGrams while reading through the file.

(b) Which method creates WordGrams by explicitly calling the Constructor every time and which one uses another method to implicitly create new WordGrams?

The benchmark method explicitly calls the WordGram constructor to create each new WordGram, while the benchmarkShift method calls the .shiftAdd() method to create each new WordGram.

2. Given that 177,634 total WordGrams were created with WSIZE = 2 but there are only 117,181 values in the Set when we add all of these WordGrams, how many duplicate WordGrams were created?

60,453 duplicate WordGrams were created

3. Find the number of duplicate WordGrams created by running WordGramBenchmark with values of WSIZE ranging from 2 to 10. That is, fill out the chart that you can see below by expanding. If you're stuck on the previous question, the chart shows you the correct answer for WSIZE = 3, which might help you work backwards to answer the previous question.

2: 60,453
3: 10,756
4: 1,987
5: 667
6: 362
7: 226
8: 151
9: 105
10: 73

4. Given what you observe in the table you filled out, Are the number of duplicates increasing or decreasing as a function of WSIZE?

As WSIZE increases, the number of duplicates decreases

5. Recall that WSIZE is the number of words in a WordGram and the WordGramBenchmark is creating WordGrams of that size/order by scanning the texts of plays. How would you explain the trend you observe that is, why is the number of duplicate WordGrams changing in the way you observe as a function of WSIZE?

The number of duplicate WordGrams decreases as WSIZE increases because WordGramBenchmark is scanning the texts for longer words, which are less likely to be duplicate than smaller, more common words like "the" or "an".

6. WordGrams like this are a useful way of breaking down a text into its constituent parts for subsequent analysis in a field called natural language processing, a subfield of artificial intelligence (or AI) and machine learning (or ML) which are parts of computer science that develop data-driven algorithms and applications. Read this article written by Duke alum Cade Metz about the people behind these technologies: Metz, C. (2021). Who is making sure the AI machines aren’t racist. The New York Times, 15.
Based on this article and your own thoughts, do you think algorithms can be biased? To what extent do you think it matters who develops algorithms? This question does not have a right/wrong answer, we are looking for a thoughtful reflection of one or two paragraphs.

I believe that algorithms can be biased, in fact it is far easier for them to be biased than not. The data that algorithms draw from is integral to the training of the algorithms and subsequent biases. If developers are to train algorithms using a subset of data that they do not confirm is representative of all individuals, the result is racially-biases in algorithms that then may be used globally, across the Internet, and even in legal instances, where racial biases are already present without the addition of AI technologies.

Because of this, I believe that when creating algorithms, it is incredibly important to involve diverse developers that may pick up on data biases that otherwise go unnoticed, particularly by white men. If the individuals working on AI development

are cognizant of how algorithms are trained, it means the scalability of these algorithms is much broader, and algorithms will be much more useful.