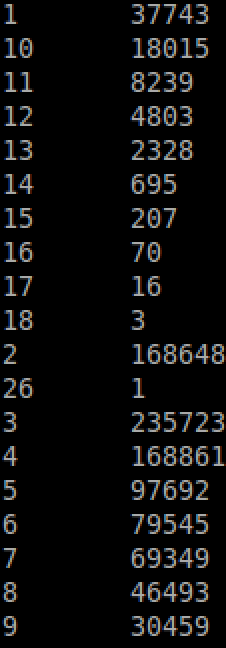
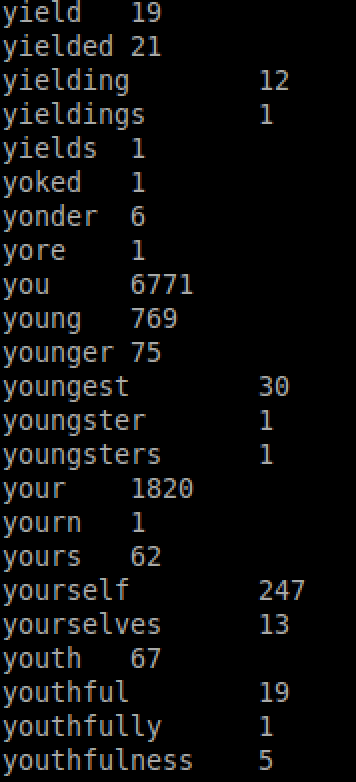
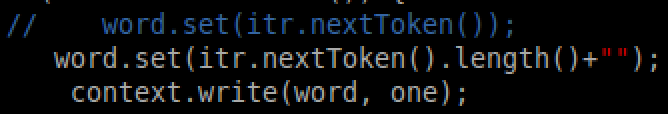
WordCount.java

Answer the following questions:

Question:Do we need to create a new method for the combiner? Explain why/why not.

Answer: We don’t need to create a new method for a combiner. We can use reducer to be our combiner. Since reducer has a function to sum up values with same key and reducer’s output type match its input type.

Question: How would you modify this program so that it counts the word-length frequencies instead?

Answer: I changed the output key of mapper. The key from mapper contained word-length information rather than word itself.

MaxWordCount.java

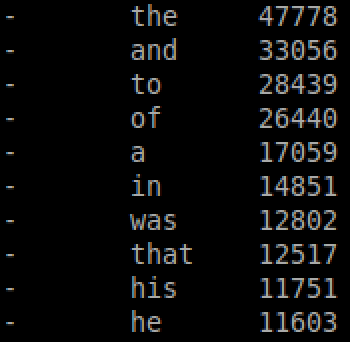
Answer the following questions:

Question: How do we ensure that all entries in the 2nd stage go to the same reducer?

Answer: Using same dummy key as the key of mapper and putting words and frequencies into value of mapper. Reducer would collect all values with same key.

Question: Do we need to create a separate combiner for the 2nd stage? Explain why/why not.

Answer: I don’t think we need to create a separate combiner for the 2nd stage. Since the output type of reducer match its input type, we can set reducer to be our combiner. I think combiner in this case should calculate a local max value and reducer need to output the global max by comparing multiple local max values. So it seems that we could use reducer as our combiner.

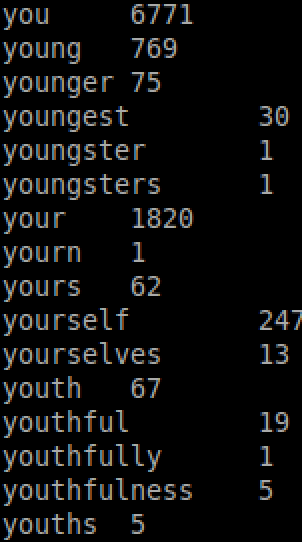
TopKWordCount.java

Answer the following question:

Question: How do Mappers and Reducers accept user-defined parameters?

Answer: Define parameters in makefile. Using the Context object to get back parameters from makefile.

WordCount2.java



Ngram.java

n=5

