Discussion Questions:

1. Describe the key benefits and consequences of your design choice
   1. Using the pipe and filter software architecture allowed me to deal with a stream of data efficiently, as well as deals with input and output nicely. I would say that the main downfall of my system is the time complexity, due to amount of for loops used within for loops to iterate through first the sentence, and the words in the sentences. If the amount of sentences provided as input were increased, then the performance of the system would slow down significantly. Perhaps if I had used a more efficient method for iterating through the sentences, this would have lead to a better design. In conclusion, the pipe and filters software architecture system allows for encapsulation of code and the pipe allows for an efficient way to deal with a stream of data, however the way I iterated through the input perhaps will affect performance in larger scales.
2. Filters in pipe and filter systems are completely independent entities.
   1. Do not share state (data) with other filters.
   2. Do not know the identity of neighbor filters in the pipeline
   3. Process their data concurrently and incrementally they do not wait for other filters to start or finish their jobs.

Would it be possible to implement an interactive version of the KWIC system, which will allow users to delete certain lines from the data flow but still not violate any of the above defined filter properties?

* + - If you tried to delete lines at runtime then this could potentially cause concurrency issues, especially since input is being received, and you are trying to delete at the same time. This would violate the third rule especially since it says that it cannot “wait for other filters to start or finish their jobs”.

1. Discuss your experiences (good or bad) with Software Architecture in your previous projects. This could include work and/or school experience. Remember that all software.
   1. I took 363 (Principles of Software Design) prior to taking this graduate level class. We discussed various software architecture patterns in that class and implemented them through programming assignments. I thought the class was an overall good experience because I learned about various techniques that could help me become a better programmer and potentially use in industry as well. I think I would have to list a bad experience it was the last project for that class, it just caused a lot of stress trying to figure out how all of the patterns fit together, but in the end, I understood how they worked together. I was weaker at programmer, but was able to strengthen my coding skills over the course of the semester. As an undergraduate student taking a graduate level course, I hope to strengthen my programming skills even more so, and become even better at it.