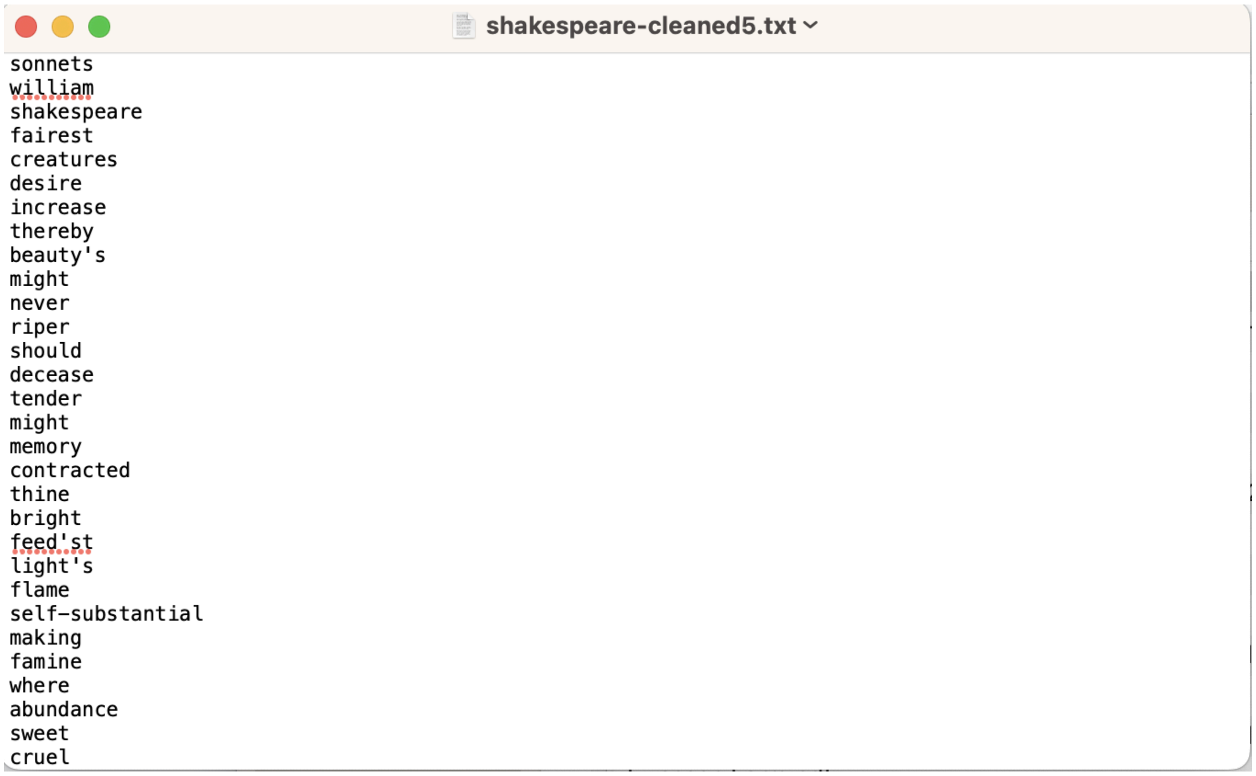
HW1 Worksheet

Problem Statement

Given a huge list of words stored in a text file (shakespeare-cleaned5.txt) , we want to find the k most frequent word of length ℓ.

Below is a screenshot of the content of shakespeare-cleaned5.txt:



```
sonnets
william
shakespeare
fairest
creatures
desire
increase
thereby
beauty's
might
never
riper
should
decease
tender
might
memory
contracted
thine
bright
feed'st
light's
flame
self-substantial
making
famine
where
abundance
sweet
cruel
```

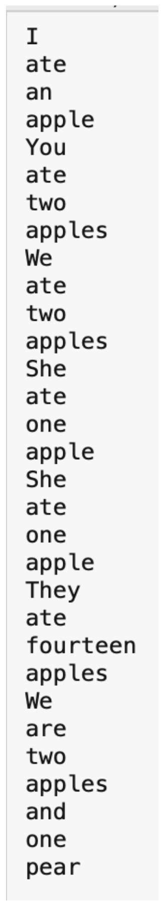
Step 1: Understand the problem (and somewhat the solution) by manually solving a toy example

Before we start to write code, we need to fully understand the problem. The best place to start is to try to solve a **toy version by hand**. In addition to helping us **understand the problem**, this will also help us come up with an **initial version of the algorithm** to solve it.

Let's make a toy (very small) version of the original problem. Let's say the composition (text) consisted of only the following sentences:

> I ate an apple. You ate two apples. We ate two apples. She ate one apple. She ate one apple. They ate fourteen apples. We are two apples and one pear.

So if we break it down to words and get rid of the periods, we get the list of words shown below.



```
I
ate
an
apple
You
ate
two
apples
We
ate
two
apples
She
ate
one
apple
She
ate
one
apple
They
ate
fourteen
apples
We
are
two
apples
and
one
pear
```

Try to solve the above toy problem **by hand on paper**. **This is not a code or pseudocode, pretend you don't know any coding**. If you change your mind, don't erase your previous writings. Just cross it in a way you can still read it AND **write down (with a different color) why you want to change it.**

The Logic/Algorithm

Please write down the algorithm in simple words (or pseudo code if you like)

Reflection:

1. Did you create any records (lists, variables, etc.) for counting/sorting? How many items does it have?
2. How many times (total) did you scan the list of words in order to do your calculations?
3. Did you use any data structure(s) ? If so, what operations should that data structure allow? (find/insert/update/delete/sort/reverse)?
4. Is your solution scalable? I.e., would it work if your input file had billions of words? Try to answer the following questions in terms of ***n*** and ***m*** where ***n*** is #total words and ***m*** is #unique words:
    a. How much RAM **memory** does your solution need? should it be a contiguous chunk (array) or not?

b. How many **times** do you need to scan the input file on the **disc** and how many **times** do you need to scan the items in your **data structure (in RAM)**?

Solve the toy problem again. This time using Linked Lists

Notice that the problem statement requires you to **use an array of Linked Lists**[1]. **Solve** the toy problem again using only linked list(s). Now you will have Linked-List related efficiency constraints to address. Draw a schematic diagram of the data structure:

Next, write down your **revised algorithm/pseudocode that only uses linked lists**:

[1] Note that using Linked Lists is not necessarily the best solution but here we want you to see the constraints that LinkedList impose (namely, traversal)

Meet (or exceed) the efficiency requirements

Read all the **instructions** carefully and **write down the efficiency constraints**. For example, given that the **queries[2] are given in a batch (in an input file)**, it is NOT reasonable to go over the Shakespeare file for every query. If your algorithm does this, you need to revise it (e.g., by trying to **load** and **organize** all the data in your data structure for the fastest possible query processing, sorting them if necessary).

Also, the grading section says that your program **should end within 2 minutes for any input file with at most 200 queries**. Does your algorithm meet this requirement? If not, you need to check where your algorithm is inefficient and come up with a solution, either by designing a different data structure, a different way of organizing/sorting the data or a different search algorithm.

Handling Special Cases and Exceptions:

Can you think of any special cases that need to be handled separately? (e.g., non-existing words, incorrect input, etc.). Note that the problem specification says that you can ignore input file errors (i.e., assume that the input file has no errors) but it requires you to handle non-existing ranks and words.

Note that this step is **VERY important. Ignoring it can have serious consequences:**

[2] Each input line (row) in the input file would be a query that asks to find the kth most frequent word of length l. For example, your input file may have 2 rows (2 queries) one of which asks for the 5th most frequent word of length 10 and the other asks for the 90th most frequent word of length 5.

- In your tests/homeworks, it will result in **failing on some test cases** and **losing points**.
- In your future **technical interviews**, it raises a **red flag** for the interviewer!

Test your code

Now create several test cases (input, correct output), including exceptional/border cases and run your code. If your code fails on any of those test cases, revise your code.

Then try the autograder which will run your code on OUR test cases.

Important Note: The test cases that we provide you with are NOT comprehensive. They are just a subset of the test cases that we use for grading your code. **This is why you need to think about all different scenarios and test your code on your own test cases as well.**

Once you are confident that your code can handle all the test cases (yours and ours), submit it according to the instructions!