# Higher Order Functions Extended Practice

Based off of section 12.5 from the book.

Goal: Create an index for a document: `makeIndex :: Doc -> Index`

Question: What types should `Doc` and `Index` have?

It makes sense to decompose `makeIndex` into a "pipeline" of steps:

```
makeIndex :: Doc -> Index
makeIndex
    = lines
    >.> numberLines
    >.> allNumberedWords
    >.> sortWords
    >.> intsToLists
    >.> groupByWord
    >.> eliminateSmallWords
```

- `lines` takes the document and splits it into a list of lines
- `numberLines` takes the list of lines and adds line-numbers to them, forming pairs.
- `allNumberedWords` replaces each numbered line with a list of number-word pairs.
- `sortWords` reorders the list of number-word pairs by word.
- `intsToLists` turns each integer into an 1-integer list.
- `groupByWord` puts together those lists corresponding to the same word.
- `eliminateSmallWords` eliminates all words of length at most 4.

1. What should be the types for each of these intermediate functions?
2. `lines` is a built-in method. What is its type? Does that match our usage of it?
3. `numberLines` is supposed to replace each line with the pair of an increasing number and the line. How can we implement that using list functions?
4. `allNumberedWords` is supposed to take each line and split it in to a list of words, then put those words together with the line's number. We can split this in steps:
   - A function that turns a line into a list of words.
   - A function that turns a numbered line into a list of numbered words.
   - A function that uses this last function to apply it to a whole list.

   Write each step.
5. `sortWords` sorts the list based on the word comparison.
   - Write a comparison for pairs, to say when a pair is "less than" another pair.

- Write a sort algorithm for pairs using that comparison. Simplest way is insertion sort: recursively sort the remaining list, then insert the remaining element in the correct spot.

6. intsToLists turns each integer into a 1-element list. You can do this via a map.

7. groupByWord needs to put together the lists corresponding to the same word. You need to actually write a function for that one, with cases for two consecutive elements having th same word.

8. eliminateSmallWords is a simple filter.