The aim of this sheet is to build a tool to help with next week’s questions on cryptography (where
you will be asked to decrypt a couple of messages that I will give you).

A histogram is a visual aid used in statistics to show the frequency distribution of particular values
in a given population. The following is an example of a crude ascii histogram showing the number
of legs of a number of sample animals:

0: ****
1: 
2: ***********************
3: *
4: ***********************

This would indicate that most of the animals studied had two legs, with a few less having four.
None had one leg, but one had three, and a few had none.

Avoid writing the type of the functions because we need to cover more material before you will
be able to do so. Also, start your script with the lines:

```haskell
import Char
import List
```

This ensures that you will be able to use a number of functions in the Haskell standard libraries
which you may need. A couple of functions which you may find useful are `unlines` and `filter`.

1. A histogram of objects can be stored as a list of pairs of objects and the number of instances
   of that object. Define a function `countOf`, which takes a histogram and an object, and
   returns how many instances of the object occurred according to the histogram. For example,
   `countOf legs histo 3` would return 1.

2. Define a function `asciiHistogram` which translates a histogram into a string representing
   the ascii histogram with one asterisk per object (as shown in the example above – 4 asterisks
   following the 0 tag indicate that 4 animals had no legs). Remember that to display the string,
   interpreting tabs and newlines, you can use the function `putStr`. You will also need to use
   the function `show` which transforms objects (of most types) into a string. Another useful
   function is `replicate`, which given an integer `n` and an object `x`, return a list consisting of
   `n` copies of `x`.

3. Define a function `sortHistogram`, which sorts a given histogram, with the more frequently
   appearing objects appearing first.

4. Define a function `top` which given a histogram and an integer `n`, truncates the histogram to
   only the top `n` appearing objects. Thus, the top 2 of the leg count example would reduce
   the histogram to one with the 0, 2 and 4 legs entries only.

5. Define a function `produceHistogram` which takes as parameters a list of objects (the sample)
   and a list of items which are to be counted, and produces a histogram object with the
   correct counts. For example, calling the function with a Shakespearian sonnet and the items
   `['a','e','i','o','u']` (or simply `"aeiou"`) would count how many times each vowel
   appeared in the sonnet.
6. Using a textbook or online notes and books, try to understand what the following function does (before executing it):

   \[\text{cleanup} = \text{filter isAlpha . map toLower}\]

Use the function on a number of English texts, and use the histogram functions you have just defined to discover the three most common letters in English.

7. The following is an example of a crude ascii histogram showing the marks students got in an exam:

   - 0-44: ************
   - 45-64: *************
   - 65-84: **************
   - 85-100: ****

This would indicate that approximately half as many students got between 85 and 100 marks as ones who scraped through with a mark ranging from 45 to 64.

In this example, we would not like to count individual objects, but to classify the objects and count instances in each of the different classes. A class can be identified as a function from the objects to a boolean value stating whether or not the object is in that class. For example, the class of marks between 45 and 64 can be defined using the classification function:

   \[\text{inRange45\_64 mark} = \text{mark} \geq 45 \land \text{mark} \leq 64\]

Define a variant of \textit{produceHistogram} which also takes a sample as parameter, but rather than the list of objects to count, it takes a list of pairs of string labels (giving the name of the class) and classification functions, to return a histogram. The marks example, would thus be called using:

   \[
   \text{results} = \text{produceHistogram2 marks}
   \]

   \[
   [("0-44", \text{inRange0\_44}), ("45-64", \text{inRange45\_64}),
   ("65-84", \text{inRange65\_84}), ("85-100", \text{inRange85\_100})]
   \]

8. Sometimes, we would like to count sequences of sub-lists in a list — for example, the number of times the “li” occurs in a given English text. Applied to the previous sentence, we would get 4. Define a function \textit{countSubList}, which given two lists, counts how many times the first appears as a substring in the other. You may find the Haskell list function \textit{isPrefixOf} to be useful.

Define \textit{pairs}, which given a list returns all pairs of symbols which occur in sequence in the list. The function \textit{nub} removes multiple copies in a list, which can be quite handy.

Combine these two functions you have just defined to generate a histogram of the 10 most commonly occurring sub-sequences of length two in a given list. Use it on a number of English texts to see whether you can identify commonly occurring pairs of letters.