28. (*expires 4/19*) The text below was encrypted with a substitution cipher (that is, a permutation). Only the letters (both upper-case and lower-case) were substituted, leaving punctuation and spaces alone; the message has proper capitalization and punctuation. Determine the original message.

```
"wA'r aBeD WUeK AP XNaB NM U rALKNP USUeAJBMA NM zUM nPrB ZNAW U JUM ZWP'r
XBUeMNMO AP SXUD AWB aNPXNM." yWUA'r ZWUA rWB APXK AWB SPXNCB ZWBM rWB WUMKBK
AWBJ AWB BJSAD eBaPXaBe.

        lNCWUeK heULANOUM, "yWB zCUeXUAAN yNXA"
```

You can find the encrypted text on the class web page at `https://www.math.stonybrook.edu/~scott/mat331.spr24/problems/extras/subscrypt.txt`. If you want to store this in a Maple string, you might want to either omit the quotation marks or prefix them with a backslash (i.e., \") when retyping them.

You do not need to write Maple that solves *every* such problem; the easiest way to do this problem involves some guesswork as well as knowledge of how English sentences are structured. After you get five or ten letters, the others should come easily.

You might find `CountCharacterOccurrences` or `CharacterFrequencies` from the library `StringTools` helpful. Depending on how you do things, `CharacterMap` could also be useful. Note that you don't really need to use Maple to do this problem, but of course you may.

29. (*expires 4/19*) In traditional "pen and paper" cryptography (that is, before the widespread availability of computers), messages were often written in uppercase letters with punctuation and spacing removed, then divided into blocks of five letters. For example, the start of this problem would be rendered as below.

```
INTRA DITIO NALPE NANDP APERC RYPTO GRAPH YTHAT ISBEF ORETH EWIDE SPREA DAVAI
LABIL ITYOF COMPU TERSM ESSAG ESWER EOFTE NWRIT TENIN UPPER CASEL ETTER SWITH
PUNCT UATIO NANDS PACIN GREMO VEDTH ENDIV IDEDI NTOBL OCKSO FFIVE LETTE RS
```

Write a Maple procedure to render a given text string this way.

Consider using `Select`, `UpperCase`, and `printf`. Also, it may be useful to remember that if the variable `msg` contains a string, one can refer to characters 10–14 of it with `msg[10..14]`.

30. (*expires 4/19*) In class on April 11, we wrote code for a Caesar cipher (or see this worksheet). This doesn't work properly when there are characters in the plaintext that aren't part of the encryption alphabet. Fix this in two different ways:

- First, write a version that just removes these offending characters when encrypting.
- Then write a version that only shifts the characters that occur in the alphabet, and replaces any other characters with an underscore character (_).

31. (*expires 4/19*) The cryptography chapter in the notes is called "`fsqFsHn sGGousG`", which is actually the result of applying a Caesar cipher to its original title. A 53-character alphabet consisting of all the upper-case letters, a space, and then all the lower-case letters was used; consequently the space in the middle might or might not correspond to a space in the title. Determine what the original title was.

The easiest way to do this is to just try all 53 possible values for the amount of the shift. The correct decryption should be obvious.