2019-11-12. **Project extended until friday**. If you aleady finished, relax! If not, get to f***ing work!

Going to implement Caesar cipher.

```
> with(StringTools) :
```

Step 1, choose the alphabet we want to write our messages in. First, let's remember the ASCII code, many characters of which are nonprinting.

```
> convert([seq(n, n = 1..127)], bytes);
"

 !"#$%'()*C ,-./0123456789:!  = O ?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}
~• "                                                              (1)
```

Folks wanted to use uppercase letters only, so let's grab those.

```
> Alphabet := Select(IsUpper, convert([seq(n, n = 1..127)], bytes))
Alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ"                              (2)
```

Want to convert a string to a list of positions in the alphabet. SearchText finds substrings in a string. (there is a friend called searchtext which ignores case, ie, treats A and a the same.)

```
> SearchText("Y", Alphabet);
25                                                                    (3)
> SearchText("YO", Alphabet);
0                                                                     (4)
> SearchText("HI", Alphabet);
8                                                                     (5)
> Message := "HELLOTHERE";
Message := "HELLOTHERE"                                               (6)
> Message[4]
"L"                                                                   (7)
```

Let's build the conversion routine step by step.

Idea: Explode the message into a list of chars, look up one by one. The ::string gives us an error if you give something other than a string as input.

```
> StringToList :=proc(str :: string) # argument has to be a string
    Explode(str);
  end:
> StringToList(Message)
["H", "E", "L", "L", "O", "T", "H", "E", "R", "E"]                    (8)
> StringToList(Pi/6)
Error, invalid input: StringToList expects its 1st argument, str, to be of type string, but received (1/6)*Pi
```

```
>
```
I would revise this as I go, but it is hard to record time, so I will keep repeating as I revise.

```
> StringToList := proc(str :: string) # argument has to be a string
    map(SearchText, Explode(str));
  end:
> StringToList(Message)
```
Error, (in StringToList) invalid arguments for searchtext

Didn't say where to search. Duh.

```
> StringToList := proc(str :: string) # argument has to be a string
    map(s -> SearchText(s, Alphabet), Explode(str));
  end:
> StringToList(Message)
```
$$[8, 5, 12, 12, 15, 20, 8, 5, 18, 5] \quad \textbf{(9)}$$

It's a good idea to mention that Alphabet is defined outside of this procedure.

```
> StringToList := proc(str :: string) # argument has to be a string
    global Alphabet;
    map(s -> SearchText(s, Alphabet), Explode(str));
  end:
> StringToList("HIHIHI")
```
$$[8, 9, 8, 9, 8, 9] \quad \textbf{(10)}$$

Now write its inverse, which converts a list back to a string
ie, get the character for each number, then squish them together.

```
> ListToString := proc(nums :: list) # argument has to be a list of positive integers.
    global Alphabet;
    Implode(map(k -> Alphabet[k], nums));
  end:
> ListToString([8, 9, 8, 9])
```
$$\text{"HIHI"} \quad \textbf{(11)}$$

At this point, we have the ability to convert strings to lists of positions in the Alphabet, and go back again.

Now let's turn to the task at hand, namely writing a Caesar cipher:
It takes a message and an amount to shift by, then
Converts the input message to a list of positions in the Alphabet.
shifts each element in the list by the given amount,
then converts back to the corresponding character string.

```
> Caesar := proc(msg :: string, shift :: integer)
      # let's allow the shift to be positive, negative, or zero.  That will be useful.
    local numlist, shifted;
    numlist := StringToList(msg);
    shifted := map(x -> x + shift, numlist);
    return(ListToString(shifted));
  end:
```

```
> Caesar(Message, 3)
```

"KHOORWKHUH" **(12)**

Looks like it worked.

```
> Caesar(Message, 20)
```

"YYY" **(13)**

Didn't work because we forgot to reduce modulo length of alphabet.
Maple lets us express modular arithmetic in two ways, either more "math like" or as a function call.

```
> 108 mod 25
```

8 **(14)**

```
> modp(108, 25)
```

8 **(15)**

So, let's change that. Note this means we need to know how long the alphabet is, so we should list it as a global.

```
> Caesar := proc(msg :: string, shift :: integer)
    local numlist, shifted, Alength;
    global Alphabet;
    Alength := length(Alphabet);
    numlist := StringToList(msg);
    shifted := map(x -> modp(x + shift, Alength), numlist);
    return(ListToString(shifted));
  end:
```

This is still broken:

```
> Caesar(Alphabet, 1)
```

Error, (in ListToString) invalid range for string subscript

```
> Caesar("ABCD", 1)
```

"BCDE" **(16)**

```
> Caesar(Alphabet, 0)
```

Error, (in ListToString) invalid range for string subscript

```
> Caesar("Z", 0)
```

Error, (in ListToString) invalid range for string subscript

```
> Caesar("Y", 0)
```

"Y" **(17)**

As you can see from the above, the trouble is with the letter Z...; since n mod 26 gives us numbers 0..25, but we want numbers 1..26

```
> 26 mod 26
```

0 **(18)**

```
> Alphabet[0]
```

Error, invalid range for string subscript

So, we have to shift by 1 before we take mod, then shift back.

```
> modp(26 - 1, 26) + 1
```

26 **(19)**

```
> modp(0 - 1, 26) + 1
```

(20)

$$26 \quad (20)$$

```
> modp(25 -1, 26) + 1
```

$$25 \quad (21)$$

```
> Caesar := proc(msg :: string, shift :: integer)
    local numlist, shifted, Alength;
    global Alphabet;
    Alength := length(Alphabet);
    numlist := StringToList(msg);
    shifted := map(x -> modp(x + shift - 1, Alength) + 1, numlist);
    return(ListToString(shifted));
  end:
> Caesar(Alphabet, 3) # shift the whole alphabet by 3
```

"DEFGHIJKLMNOPQRSTUVWXYZABC" **(22)**

```
> Caesar(%,-3) # now shift back.
```

"ABCDEFGHIJKLMNOPQRSTUVWXYZ" **(23)**

Let's try with a different Alphabet.

```
> Alphabet := Select(IsPrintable, convert([seq(n, n = 1..127)], bytes))
```

Alphabet := **(24)**

" !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
^_`abcdefghijklmnopqrstuvwxyz{|}~"

```
> Caesar(Message, 47)
```

"wt{{~$wt"t" **(25)**

Since we have a different alphabet, shifting the upper case message pushes a few characters into the symbols that live at the end ({, }, and ~) or at the beginning ($ and ").
We can also do messages with mixed case and punctuation:

```
> Caesar("I have heard the mermaids singing, each to each, I do not think they
      will sing to me.", 47)
```

"xO81F5O851B4OD85O=5B=194COC9O79O7[O5138OD?O5138[OxO4?OO? **(26)**
DOD89O;OD85IOG9! ! OC9O7OD?O=5]"

```
> Caesar(%,-47)
```

"I have heard the mermaids singing, each to each, I do not think they will sing **(27)**
to me."

Let's go back to the lowercase alphabet.

```
> Alphabet := Select(IsUpper, convert([seq(n, n = 1..127)], bytes))
```

Alphabet := "ABCDEFGHIJKLMNOPQRSTUVWXYZ" **(28)**

Note that breaking the Caesar cipher is trivial...

```
> Crypto := Caesar("SECRETSTUFF", 12)
```

Crypto := "EQODQFEFGRR" **(29)**

Probably you can just guess, but another way is just list all the possibilities. Only one of them is actual English, so easy.

```
> for i from 0 to length(Alphabet) do
```

```
  print(i, Caesar(Crypto,-i));
od;
```

```
                              0, "EQODQFEFGRR"
                              1, "DPNCPEDEFQQ"
                              2, "COMBODCDEPP"
                              3, "BNLANCBCDOO"
                              4, "AMKZMBABCNN"
                              5, "ZLJYLAZABMM"
                              6, "YKIXKZYZALL"
                              7, "XJHWJYXYZKK"
                              8, "WIGVIXWXYJJ"
                              9, "VHFUHWVWXII"
                             10, "UGETGVUVWHH"
                             11, "TFDSFUTUVGG"
                             12, "SECRETSTUFF"
                             13, "RDBQDSRSTEE"
                             14, "QCAPCRQRSDD"
                             15, "PBZOBQPQRCC"
                             16, "OAYNAPOPQBB"
                             17, "NZXMZONOPAA"
                             18, "MYWLYNMNOZZ"
                             19, "LXVKXMLMNYY"
                             20, "KWUJWLKLMXX"
                             21, "JVTIVKJKLWW"
                             22, "IUSHUJIJKVV"
                             23, "HTRGTIHIJUU"
                             24, "GSQFSHGHITT"
                             25, "FRPERGFGHSS"
                             26, "EQODQFEFGRR"                              (30)
```

Clearly the message was "SECRETSTUFF" encoded with a shift of 12.

Here's a bug. (Actually, its not a "bug" per se, but an issue)

```
> Caesar("This does NOT work as it should!", 6)
```

```
                  "ZFFFFFFFFFTUZFFFFFFFFFFFFFFFFFF"                       (31)
```

```
> Caesar(%,-6)
```

```
                "TZZZZZZZZZNOTZZZZZZZZZZZZZZZZZZ"                         (32)
```

Homework. Fix it.