Last time we wrote a function that takes a message and changes it into a number by interpreting the alphabet numberically as A=0, B=1, ... and reading the word as a number written in base 26. This has the drawback of being a non-injective function from words to numbers, meaning that more than one word is represented as the same number.

For example, try running your code on the words `'RDVARK', 'ARDVARK', 'AARDVARK'` ; you should get the same number for each (203723868). The issue here is that since A=0, the additional A's at the beginning have the effect of adding 0 to the number.

$$\begin{aligned} RDVARK &\sim 17 \cdot 26^5 + 3 \cdot 26^4 + 21 \cdot 26^3 + 0 \cdot 26^2 + 17 \cdot 26 + 10 \\ AARDVARK &\sim 0 \cdot 26^7 + 0 \cdot 26^6 + 17 \cdot 26^5 + 3 \cdot 26^4 + 21 \cdot 26^3 + 0 \cdot 26^2 + 17 \cdot 26 + 10 \end{aligned}$$

To circumvent this issue we are going to break our messages up into pieces of a fixed size. When changing from numbers back into words, we can check if each piece of the word is long enough, and if not we add extra A's to the beginning.

1 Blocking up a message

We want to write a function that takes a message, breaks it into blocks of a fixed size and then returns a list of numbers corresponding to these blocks interpreted as base 26 numbers. In order for this to work, we need the size of the blocks to evenly divide the length of the message. This is a serious inconvenience, and we get around it by saying that if the blocksize does not divide the length of the message then add extra X's to the end of the message so that it is.

I may call my function `blockstring(message,blocksize)`. If I run this function on the inputs `('CHEMICALWARFARE',2)` I want the function to essentially do the following:

```
CHEMICALWARFARE

CHEMICALWARFAREX

[CH,EM,IC,AL,WA,RF,AR,EX]

[59, 116, 210, 11, 572, 447, 17, 127]
```

Notice that `AL` encodes to 11, which is the same thing that just `L` would encode to. We do not have to worry about this at the present time.

Similarly, if I run this function on the intputs `('CHEMICALWARFARE',3)` my function should essentially do the following:

```
CHEMICALWARFARE

[CHE, MIC, ALW, ARF, ARE]

[1538, 8322, 308, 447, 446]
```

Remember that Sage has the ability to pull out part of a string. For example, if I define `message='CHEMICALWARFARE'` then the command `message[0:2]` returns `'CH'` and `message[2:4]` returns `EM`.

Now we want to write a function `stringblock(message,blocksize)` that reverses the above process. Here is where we need to be careful about adding in extra A's when needed. What we do is run each of the numbers in the input list through our function that takes a number to a word, then we check the lengths of each of these and if they are not the indicated blocksize we include extra A's at the beginning.

For example, running this function on the input `([1538, 8322, 308, 447, 446],3)` should do essentially the following:

```
[1538, 8322, 308, 447, 446]

[CHE, MIC, LW, RF, RE]

[CHE, MIC, ALW, ARF, ARE]

CHEMICALWARFARE
```

Something to be careful about is that sometimes you may need to add in more than one A. For example, if you want to turn the blocked message `[227474, 213842, 456, 244657]` back into text, and you know that the blocksize is 4, then running `stringblock([227474, 213842, 456, 244657],4)` should do the following:

```
[227474, 213842, 456, 244657]

[MYNA, MEIS, RO, NXXX]

[MYNA, MEIS, AARO, NXXX]

MYNAMEISAARONXXX
```

2 ElGamal

Let us recall how ElGamal works. Alice publishes a public key `(p,g,X)` where `X` is some power of `g` mod `p`; `X` $=g^a$`mod p` where `a` is a secret number Alice chose. You have a message `m` that you want to send to alice. You choose any number `b`, compute `k` $=X^b$`mod p` and `Y` $=g^b$`mod p` and encode your message by `e = m*k mod p`. You then send Alice the pair `[e,Y]`; `e` being the encrypted message and `Y` what Alice needs to decrypt it.

- Write a function `ElGamal(message,p,g,X,b,blocksize)` that takes in the public key information and your chosen secret number and outputs the encrypted message with the decryption key. To encrypt the message, use your `blockstring(message,blocksize)` function to break turn the message into a list of numbers and do the encryption on each of these numbers.

  For example, if the public key is `(93059,10,57405)` and I choose my private number to be b = 321 then running `ElGamal('CHEMICALWARFARE',93059,10,57405,321,3)` does essentially the following:

  ```
  k = 57405^321 mod 93059

  Y = 10^321 mod 93059

  CHEMICALWARFARE

  [1538, 8322, 308, 447, 446]

  e = [1538*k mod 93059, 8322*k mod 93059, 308*k mod 93059, 447*k mod 93059, 446*k mod 93059]

  e = [2896, 52579, 56851, 22382, 21291]

  return [e,Y]
  ```

  Now pretend your Alice and you have to decrypt a message someone sent you using your public key. You raise the `Y` you received to the power of your secret number `a` and reduce mod `p` to get the `k` that was used to encrypt; `k` $=Y^a$`mod p`. You then compute the inverse of `k` mod `p` and multiply the message you recieved by this inverse to decrypt it.

- Write a function `unElGamal(e,p,Y,a,blocksize)` that unencrypts a message. Continuing the above example, if your public key was `(93059,10,57405)` then your secret number was `a=123`. Bob sends you the encrypted message `[[2896, 52579, 56851, 22382, 21291], 5323]` and tells you the block-size used for encyption was 3. To decrypt this you run `unElGamal([2896, 52579, 56851, 22382, 21291],93059,5323,123,3)` and the following, in essence, occurs:

```
k = 5323^123 mod 93059

inversek = modInv(k,93059)

[2896, 52579, 56851, 22382, 21291]

[2896*inversek mod 93059, 52579*inversek mod 93059, 56851*inversek mod 93059, 22382*inversek mod 93

[1538, 8322, 308, 447, 446]

CHEMICALWARFARE
```

- If you get your code working, go to `http://summermathprogram2016.blogspot.com/` where I've posted a public key. Use this to post a coded message in the comments section and I'll decrypt it. Post a public key of your own and I'll send you a message to decrypt.

3 Euler totient function

Recall the totient function $\varphi(n)$ is defined to be how many numbers smaller than n are relatively prime to n. We have seen that this function satisfies the following two equations:

$$\varphi(p^k) = p^k - p^{k-1} \qquad\qquad \varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$$

In these equations, p is a prime number and m and n are relatively prime. Using these two equations we get the following method to compute the totient function: given any number n we determine it's prime factorization $n = p_1^{k_1} p_2^{k_2} ... p_m^{k_m}$ and then

$$\begin{aligned} \varphi(n) &= \varphi(p_1^{k_1} p_2^{k_2} ... p_m^{k_m}) \\ &= \varphi(p_1^{k_1})\varphi(p_2^{k_2}) \cdots \varphi(p_m^{k_m}) \\ &= (p_1^{k_1} - p_1^{k_1-1}) \cdot (p_2^{k_2} - p_2^{k_2-1}) \cdots (p_m^{k_m} - p_m^{k_m-1}) \end{aligned}$$

- Write a function `totient(n)` that computes the Euler totient function in the above manner. You will definitely want to make use of an auxiliary function that computes the prime factorization of a number, counting repeats.

4 Bonus Questions

- Observe $2+3+5+7+11+13=41$. Notice that 2; 3; 5; 7; 11; 13 are consecutive primes and that 41 is also prime. Find the largest number below 1000 that is prime and is the sum of consecutive primes.

- Every prime number is odd (besides 2) and every odd number can be written as $4n+1$ or $4n-1$ for some n. Consequently, every prime number (besides 2) can be written as $4n+1$ or $4n-1$ for some n. The odd primes smaller than 10 are 3,5,7 ; notice that two of these are of form $4n-1$ and one is of form $4n+1$.

  If we consider all of the odd primes smaller than 50, we get the following list,

  $$3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47$$

  Notice that there are eight primes on this list of form $4n-1$ and six of form $4n+1$.

  In both of these examples, when we look at all of the prime numbers below a certain number there are more of the form $4n-1$ than of form $4n+1$. However this rule does not always hold; find the smallest k such that there are more primes smaller than k of form $4n+1$ than of form $4n-1$.