RSA cryptography instructions

1. **Generate your public and private keys**: $n = 32$ the number of digits

   (a) Pick p, q primes bigger than 10^n, so that your moduli will be bigger than 10^{2n}. This is still not big enough to be secure, but you will be able to send messages with up to $2n$ digits (so n letter/punctuation symbols) per packet.

   ```
   import primes
   n=32
   primes.randprime(n)
   ```

   (b) Calculate $N = pq, N_2 = (p-1)(q-1)$.

   (c) Find the encryption power e:

      i. Pick a random number r between 10^n and 10^{n+1}.

      ```
      import random
      random.randint(10**n,10**(n+1))
      ```

      ii. Check if r and N_2 are relatively prime (so that the inverse exists), i.e check $\gcd(r, N_2) = 1$.

      ```
      import fractions
      fractions.gcd(r,N2)
      ```

      iii. If $\gcd(r, N_2) = 1$, then $e = r$. Otherwise go back to i.

   HINT: You can use a for or while loop to make this step faster.

   (d) Find the decryption power d so that $de \equiv 1 \mod N_2$, this can be done with

   ```
   primes.modinv(e,N2)
   ```

2. **Check your RSA keys** with `primes.rsa_check(p,q,N,N2,e,d,n)`. To **publish your public key pair** (N, e), go to

   `http://rush.math.utah.edu:8080/`

   and follow the instructions.

   Please do not continue until your public key has been published in the class website! Published public keys will also be available from this website.

3. **Create a secret message** that is not longer than 96 characters (including space and special characters). Say the message is in a variable `m` as a string.

   (a) Use `primes.chop_message(m,n)` to check that you don't have more than 3 packets.

   (b) Use `primes.encode_message(m,n)` to both encode the message with Davis table and chop it into packets. The Davis table simply replaces each character using the following table:

	0	1	2	3	4	5	6	7	8	9
0	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX
1	SP	A	B	C	D	E	F	G	H	I
2	J	K	L	M	N	O	P	Q	R	S
3	T	U	V	W	X	Y	Z	a	b	c
4	d	e	f	g	h	i	j	k	l	m
5	n	o	p	q	r	s	t	u	v	w
6	x	y	z	.	,	:	;	’	”	‘
7	!	@	#	$	%	^	&	*	−	+
8	(	)	[	]	{	}	?	/	<	>
9	0	1	2	3	4	5	6	7	8	9

   For example, “U” is 31 and “5” is 95. The entries with XX are not used, and “SP” is the space character. Thus all characters code to between 10 and 99.

   (c) Make sure no packet starts with 0.

4. **Write an encryption and decryption function**: $\text{encrypt}(x) = x^e \mod N$, $\text{decrypt}(x) = x^d \mod N$.
   To define a function $f(x) = 2x$:

```
def fct(x): return 2*x
# alternatively you could write
fct = lambda x: 2*x
```

   The following computes $a^b \mod N$:

```
pow(a,b,N)
```

5. **Create a digital signature**:

   (a) Create a plaintext signature which identifies your group, using at most 32 characters. Convert your signature into a number with at most 64 digits, using Davis' table. Let `s` be the signature string.

```
primes.davis_enc(s)
```

   (b) We are using the secure signature feature, so decrypt your signature using your own (secret) decryption power d. This will create a long sequence of digits, a number less than your groups' modulus but probably with 64 or 65 digits and potentially larger than the moduli of groups you're sending to. This means you will need to break your decrypted signature into two packets of at most 64 digits each (and so that the second one doesn't start with 0).

```
# transform number sig into a string and chop it into
# packets of at most 2*n = 64 digits
sigs = primes.chop_message(str(sig),2*n) # gives strings
int(sigs[0]) # to convert the first packet into an int
```

6. **Encrypt and send your message**. You should have up to 5 packets of at most 64 digits: 3 from your plaintext secret message and 2 from your decrypted signature. None of these packets should have a leading 0.

   (a) If you are group x then you will be sending messages to groups $x+1$ and $x-1$ mod 7 (where, of course, we rename group 7 as group 0).

   (b) Encrypt the (up to) 5 packets, using the public encryption keys for the two groups you're sending them to.

```
# here is nifty way of applying a function f to
# each element in a list l:
fl = map(f,l)
# and here is an example that computes the squares
# of the numbers 1 through 5
def f(x): return x**2
fl = map(f,[1,2,3,4,5])
```

   (c) Post your packets to the following webpage:

`http://rush.math.utah.edu:8081/`

   Use the following python friendly format (origin/destination of the message, and what the packets are)

```
# this is a message from group 1 to group 2
sender = 1
dest   = 2
p1 = 12983198319038120938
p2 = 19283192831983192831
```

```
p3 = 23129748313740088347
s1 = 12331231231231231231
s2 = 99123992399123188844
```

7. **Decrypt the message**.

   (a) Use your private key and the decrypt function to decrypt the two messages you receive.
   Again you can use `map(f,l)` to apply the function `f` to each element of the list `l`.

   (b) Separate the first three blocks (text) and last two blocks (signature).

   (c) Decode the first three blocks to get the original plain text,
   use `primes.decode_message(m)`

   (d) Glue the last two packets together, transform them into an integer (which will be 64 or 65 digits long) and encrypt this integer with the sender's public key. Decode the signature to the original plain text.