RSA cryptography instructions

1. Here are the modules we will be using:

```
from primes import *
import random,fractions
```

2. **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.

   ```
   n=32
   p = 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}.

      ```
      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: Use a for or while loop.

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

   ```
   modinv(e,N2)
   ```

3. **Check your RSA keys** with `rsa_check(p,q,N,N2,e,d,n)`. To **publish your public key pair** (N, e), send an email to the instructor using the following Python friendly format, replacing the numbers by your own. **Please send this as plain text, and not as an attachment**

```
group_number = 1 # your own group number
names = [ "FirstName1", "FirstName2" ]
p = 23
q = 41
N = 943
N2 = 880
e = 7
d = 503
```

   This will allow the instructor to double check that all the numbers you are working with are OK and to publish **only the public key part** (N, e) for the other groups to use. The keys will be available on the website as a list of tuples (N, e) for each group, where the last group is renamed as group zero. For example, if we have 4 groups:

```
 # assuming 4 groups
 pubkeys = [
    # group 4
    (1983129381923,123123),
    # group 1
    (1231231231231,123123),
    # group 2
    (1231231231231,123123),
    # group 3
    (1231231231231,123123)
   ]
```

   In this way to get the N and e from from number i we can do `Ni,ei = pubkeys[ i%4 ]` (assuming 4 groups).

   Please do not continue until your public key has been published in the class website!

4. **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 `encode_message(m,n)` to both encode the message with Davis table and chop it into packets (of $2n$ digits at most each)

   (b) Make sure you get no more than 3 packets.

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

   ```
   def fct(x): return y=2*x
   ```

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

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

6. **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.

   ```
   s_int = davis_encode(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. Let's assume the decrypted signature (an integer) is in variable `s_signed`:

   ```
   s_chopped = chop_integer(s_signed,64)
   ```

7. **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 4 (where, of course, we rename group 4 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) Email your packets to the instructor, so that they are posted in the website . Use the following format (to make things Python friendly and to the origin/dest of the message, and what the packets are). **Please send this as plain text, and not as an attachment**

```
# this is a message from group 1 to group 2
sender = 1
dest   = 2
p1 = 12983198319038120938
p2 = 19283192831983192831
p3 = 23129748313740088347
s1 = 12331231231231231231
s2 = 99123992399123188844
```

8. **Decrypt the message**.

   (a) Use your private key and the decrypt function to decrypt the two messages you received. 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). If `l` is a list with 5 elements, `l[0:3]` is the list with only the first 3 elements and `l[3:]` is the list with the last two elements.

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

   (d) Glue the last two packets together (use `join_integer`), this will give you an integer (which will be 64 or 65 digits long). Encrypt this integer with the sender's public key. Decode the signature to the original plain text (use `davis_dec`).