RSA Implementation Homework

Due dates.
Public keys emailed to me by Thursday April 12.
I will send to you the ciphertexts no later than Saturday April 14.
You will then send back to me the decoded text by Tuesday April 17.

A) Overview
You should already have programmed square and multiply (modular exponentiation) and finding inverses mod *n* using the Extended Euclidean Algorithm.
First Program the Miller-Rabin prime testing Algorithm 5.7.

You will generate the keys for the RSA algorithm, *p*, *q*, *a,* and *b*. You will email the public key (*n*, *b*) to me. I will encrypt coded text. You will decrypt using the private keys and decode into text. You will then send the decoded text back to me. I will provide the Python code to decode the text using Python 2.7.

B) Detailed Steps:
1) Generate two big prime numbers *p* and *q*, each about 512 bits using the next two steps to generate each.
2) Generate a 512 random number using *s* = random.getbits(512); if *s* even add 1.
3) Test if *s* is prime using the Miller-Rabin prime test; *s* is prime if Miller-Rabin(*s*) runs 20 consecutive times with only prime results. If *s* is not prime go back to step 2. Do this two times to find *p* and *q.*
3) Calculate phi = $(p - 1)(q - 1)$ and $n = pq$.
4) Generate a random *b* or use the popular prime $2^{16}+1 = 65537$ as *b*. Test if gcd(*b*, phi) = 1. If not, choose another *b*. Calculate *a* as the inverse of *b* mod phi.
5) The RSA public key is (*n*, *b*). Make sure to save your private key, it will be needed later to decrypt!

C) Sending the public key to me
Save the keys to a text file using the following Python:
Note xxxxx is your last name. Remember to save your keys, you will need them to decrypt

```
keyFid = open('xxxxxx_keys.txt','w');
keyFid.write(str(n))   #RSA n
keyFid.write('\n')
keyFid.write(str(b))   #RSA b
keyFid.close()
```

I will read the keys and convert back to an integer using:
n = int(keyFid.readline()) and
b = int(keyFid.readline()).

I will then encode and encrypt some text using your public key and mail the ciphertext back to you in a text file. You will read them using:

```
keyFid = open('xxxxxx_cipher.txt','r');
y = int(keyFid.readline())
keyFid.close()
```

D) Decrypt
You will then decrypt the RSA cipertext $x = d_k(y)$. You can convert the big integers back into strings using the supplied function int2str():
RSAstr = int2str(x)

Email back to me the decrypted strings using:
keyFid = open('xxxxxx_text.txt','w');
keyFid.write(RSAstr) #RSA string
keyFid.close()

E) Code to convert big integers to strings and vice versa.

```
#Convert a string to a big integer
def str2int(s):
 return int(s.encode('hex'), 16)

#Convert a big integer to a string
def int2str(i):
 h = hex(i)
 return hex2str(h)

#Helper function for int2str()
def hex2str(h):
   if len(h) > 1 and h[0:2] == '0x':
     h = h[2:]

   if h[len(h)-1] == 'L':
      h = h[0:len(h)-1]

   if len(h) % 2:
      h = "0" + h

   return h.decode('hex')
```