Bonus Homework 1

Implementation:

To replicate the RSA schema discussed in class, the storage of binary strings a vectors plays a crucial role, similar to prior homework assignments. Many tools and portions of code were developed as early as program 2. To begin, a class was created for binary manipulations; this includes (but is not limited to) binary arithmetic, modular exponentiation and binary/decimal conversions. Using this class, the algorithm is laid out as by the book.

Initially the user is prompted to input a binary string which the program will encode, then an integer value, *n*, which denotes the number of bits the primes *p* and *q* are in length. The binary string is then translated from a string to integer vector by a simple function `stringToVec`. *p* and *q* are generated by a function based on Homework 3, which generates a prime number, given a number of bits. Curious as to what may happen if *p* and *q* are equal, the decision was made to ensure *p* != *q* in case this were to be detrimental to encryption. *N* was then computed via another previously developed function, `BinMultiply`. Last in the simple computations was the encryption key, *e*, which was generated as a prime number, which was also relatively prime to *(p-1)(q-1).* Since the developed number was already prime, the only factor which could make it **not** relatively prime to *(p-1)(q-1)* would be if *(p-1)(q-1)* was divisible by e.

Most of the work in composing this program came from the function which finds *d*, the private key which is the inverse of e with respect to *(p-1)(q-1).* The coefficients *a* and *b* had to be determined in the equation:

$$a \cdot (p-1)(q-1) + b \cdot e \equiv 1$$

if b was positive, *d* = b, and if b was negative, *d* = *b+(p-1)(q-1)*. This entails the following general idea:

$e \cdot d \equiv 1 \, mod \, (p-1)(q-1)$ so that $(x^e)^d \equiv x \, mod \, N$. The method of finding *a* and *b* were similar to the demonstrated method in class (on paper), but utilizing a vector of binary division results, containing both quotients and remainders per step. (The binary division function `BinDiv` is an improved method of the division function seen in Homework 3).

Lastly, the binary message was encoded utilizing the `modexp` function, as:

encoded message = $x^e \, mod \, N$, where x is the binary message. The encoded message is output to the screen, then the decoding process begins. To decode, the encoded vector is passed into `modexp` along with the decryption key and N. decoded message = $(encoded)^d \, mod \, N$. The result is then output to the screen, to no surprise the result matches the input binary message!

CPU Time:

Bits	Time (seconds)
6	5
32	111
64	1653
128	N/A
256	N/A

**256-bit encryption was attempted and allotted a copious amount of time to compute. After approximately 10 hours on raw computation the program seemed to have made little progress and was terminated by choice. No results were obtained for the 256-bit encryption case, this is most likely subject to coding mistakes or inefficiencies supplied by myself.

Estimated Time:

(General estimates, subject to correction)
Computing:
p, *q*, *e*: O(n^3) (determined in Homework 3)
N (multiplication): O(n^2)
(p-1)(q-1) (subtraction and multiplication): O(n) + O(n^2)
d (finding a and b): log(n)
Encode, decode (modexp): O(n^3) (stated in the book)

By the growth between the three time values found, I would believe the running time relation could be similar to the following: $n^2 + \text{n} + \log(n)$ at least... if not greater.

*This page does not have to be returned with graded report. Probably a waste of paper.

Testing Results:

(as output by the program)

The data below is shown in decimal, as it is output to the screen in decimal, but the data being manipulated throughout the entire program is in binary. The conversion to decimal is just easier to read when testing.

32-bit

```
Input the binary message to encode: 1001101011
Input number of bits for primes: 32
Binary p: 10011010111111010110111101010011
Binary q: 10010000001110000111011011101111
In Decimal: 2600300371  2419619567
Decimal N: 6291737657748959357
k: 63
p - 1: 2600300370
q - 1: 2419619566
(p-1)(q-1) :  6291737652729039420
e: 19
a is: 6
b is: -1986864521914433501
PUBLIC KEYS: (N, e) = 6291737657748959357, 19
d is: 4304873130814605919
encoded message: 101001110010110011110111001001001011100011010100100010000001100
decoded message: 1001101011
Press any key to continue . . .

Input the binary message to encode: 11001101011
Input number of bits for primes: 32
Binary p: 10101100010001010110010011000011
Binary q: 11001100111100110011110000010011
In Decimal: 2890228931  3438492691
Decimal N: 9938031054560243321
k: 64
p - 1: 2890228930
q - 1: 3438492690
(p-1)(q-1) :  9938031048231521700
e: 17
a is: 1
b is: -584590061660677747
PUBLIC KEYS: (N, e) = 9938031054560243321, 17
d is: 9353440986570843953
encoded message: 10010101000010111101010111001100110111011011011000110100010111
decoded message: 11001101011
Press any key to continue . . .
```

64-bit

```
Input the binary message to encode: 1001101011
Input number of bits for primes: 64
Binary p: 1011010011000000110111010001101011100100000001011010111000101011
Binary q: 1100000101001111111011000011111011000110101110111110011010100111
In Decimal: 13024653229919940139       13929611926846564007
Decimal N: 181428364974533420813527014527271976973
k: 128
p - 1: 13024653229919940138
q - 1: 13929611926846564006
(p-1)(q-1) :  181428364974533420786572749370505472828
e: 29
a is: -9
```

b is: 56305354647268992657901887735674112257
PUBLIC KEYS: (N, e) = 181428364974533420813527014527271976973, 29
d is: 56305354647268992657901887735674112257
encoded message: 1000011011010101001111011001011000010011000100011101001001111111
100001010011000010111010111101100101101111101001101110111100011
decoded message: 1001101011
Press any key to continue . . .

Input the binary message to encode: 11001101011
Input number of bits for primes: 64
Binary p: 1011100010101010001001101000011101010100100100010101010110011001
Binary q: 1000110011010101000000111101101001011000011100000000100000011011
In Decimal: 13306490411700802969 10148021571670771739
Decimal N: 135034551741170037040876217854212493091
k: 127
p - 1: 13306490411700802968
q - 1: 10148021571670771738
(p-1)(q-1) : 135034551741170037017421705870840918384
e: 17
a is: -7
b is: 55602462481658250536585408299758025217
PUBLIC KEYS: (N, e) = 135034551741170037040876217854212493091, 17
d is: 55602462481658250536585408299758025217
encoded message: 1100100000010000100010001000000000110101110111011001100010001000
1110010011000101011111010010000111011011001110110011010101010001
decoded message: 11001101011
Press any key to continue . . .

128-bit
Input the binary message to encode: 1001101011
Input number of bits for primes: 128
Binary p: 10111100000100001010011101111111110001001100001001101000111011010001010
0000001110110100001001111101101011000100100100000110111111
Binary q: 1011100101000100011101011010101101110000010101010111011001100100001000
0010011110000010001101010000110011101101100100001101010111

In Decimal: 249981337224241206041088895930212696511 246262642031294860831513
313155810214743

Decimal N: 615610645633577170038531391603729324914263820104689792869643699280439
96861673

k: 256

p - 1: 249981337224241206041088895930212696510
q - 1: 246262642031294860831513313155810214742
(p-1)(q-1) : 615610645633577170038531391603729324909301380312134432200917677189
57973950420

e: 31

a is: 4
b is: -7943363169465511871464921181983604192378082326608186221947324866962319219
409

PUBLIC KEYS: (N, e) = 6156106456335771700385313916037293249142638201046897928696
4369928043996861673, 31

d is: 53617701393892205132388217978389328298552055704605256998144442851995654731
011

encoded message: 1101000111000000010011100001100011101010011100010110000111111111
01101100001111011011111000010011010100111101010111010100011101111100011111101110
10101110011100110001111000110011001001110011100100000100101000011000001011001010
1001100110011001100001110011111

decoded message: 1001101011

Press any key to continue . . .

Input the binary message to encode: 11001101011
Input number of bits for primes: 128
Binary p: 1101011001001001011000011101000001111011100110111011001101110110110001
11110001010101010010100101011010001100101001011000000011011
Binary q: 1110001001001000110110010100100110110010000011001001111001111001001010
11000101001111001001101101111101000100000110101001101001 01

In Decimal: 284835812680089283602087729665355175963   300783779542848938046392
728944434588069

Decimal N: 8567399228707619121020639161818520040071574255269907247074274389519 29
15385447

k: 256

p - 1: 284835812680089283602087729665355175962
q - 1: 300783779542848938046392728944434588068
(p-1)(q-1) : 856739922870761912102063916181852004001301229604761342490942634365
83125621416

e: 23

a is: -4
b is: 1489982474557846803655763332490177398263132573225671899984248059766663054 2
855

PUBLIC KEYS: (N, e) = 85673992287076191210206391618185200400715742552699072470 74
274389519291538 5447, 23

d is: 1489982474557846803655763332490177398263132573225671899984248059766663054 2
855

encoded message: 1100111001001111111011101101101011101110011011100010011011 00011
1110011110001111001010001110011101101000000000100001011001011010111111101001101 1
1111001000000100000100110100001100100111001110011011000011101001001000001000101 1
1001000100010011010 0011

decoded message: 11001101011

Press any key to continue . . .