Name: Lawrence Bush
RCS Account: bushl2

Section: 1
Email: bushl2@rpi.edu

TABLE OF CONTENTS

INTRODUCTION

This program implements a cryptography system.
It consists of three parts. The first part creates a key,
the second part encodes a message, and the third part decodes
the message.

The program should be compiled and run in three separate workspaces.
An organized way to do this, is to put the files into three separate
folders as follows.

```
create key folder:
      createkey.cpp
      gcd.h
      mmi.h

encode folder
      encode.cpp
      modexp.h

decode folder
      decode.cpp
      modexp.h
```

PURPOSE OF EACH SOURCEFILE:

createkey.cpp	-	This is the main file for the create key workspace. The program accepts two prime numbers and computes a key (e), a decode key (d) and the product (N) of the inputs.
encode.cpp	-	This is the main file for the encode workspace. The program accepts three arguments (M,e,N) (Message, encode key, product of the two primes). It then uses this information to encode M.
decode.cpp	-	This is the main file for the decode workspace. The program accepts three arguments (R,d,N)

```
                    (Encoded Message, decode key, product of the
                    two primes).  It then uses this information to
                    decode N.

      gcd.h         -     This header file contains a function which
                    computes the greatest common divisor of two
                    inputs.

      mmi.h         -     This header file contains a function which
                    computes the Modular Multiplicative Inverse.

      modexp.h      -     This header file contains a modexp function
                    which recursively calls itself to calculate
                    and return X^N mod P.
```

COMPILE AND RUN INSTRUCTIONS:

CREATE KEY

Open Create Key folder.
Open creatkey.cpp in MS Visual C++
Compile and run(! button)

The .h files (gcd.h, mmi.h) should be in the same directory as creatkey.cpp. I always add the .h files to the project, but it is not necessary.

Command line arguments:

Note: If you input the command line arguments incorrectly, the program will instruct you on how to input them.

The create key program takes two prime numbers as command line arguments.
They are used in the program as the variables: [p, q].
For example: [5, 11]

The program outputs an encode key [e], a decode key [d] and the product of the two primes [N].

ENCODE

Open encode folder.
Open encode.cpp in MS Visual C++
Compile and run(! button)

The .h files (modexp.h) should be in the same directory as encode.cpp. I always add the .h files to the project, but it is not necessary.

Command line arguments:

Note: If you input the command line arguments incorrectly, the program will instruct you on how to input them.

The encode program takes three numbers as command line arguments.
They are Message, encode key, and the product of the two primes used to develop key [M, e, N]. For example: [47, 3, 55].
e and N are outputs from create key.

The program outputs an encoded message [R].

DECODE

Open Decode folder.
Open decode.cpp in MS Visual C++
Compile and run(! button)

The .h files (modexp.h) must be in the same directory as decode.cpp. I always add the .h files to the project, but it is not necessary.

Command line arguments:

Note: If you input the command line arguments incorrectly, the program will instruct you on how to input them.

The decode program takes three numbers as command line arguments.
They are Encoded Message (output by encode program), decode key, and the product of the two primes used to develop key [R, d, N].
For example: [38, 27, 55].
d and N are outputs from create key.

The program outputs an decoded message [Q], which should be the same a the origonal message [M].

RESULTS -

I tested the create key program using the two primes: 5 11

The results for are:

```
p = 5, q = 11
N = 55
Public Key:  e = 3
Private Key: d = 27
```

These outputs are correct because:

```
N = p * q = 55

gcd(e,(p-1)*(q-1)) = gcd(3,40)= 1

d = 27 = mmi(e,(p-1)*(q-1))
e*d = 1 (mod ((p-1)*(q-1)) )
3*27 = 1 (mod 40)
81 % 40 = 1
```

I tested the encode program using: 47 3 55

```
The message to be encoded is 47.
The public (encoding) key is 3.
The N is 55.

The results are:

M = 47, e = 3, N = 55
Encrypted Message R = 38
```

This output is correct because:

```
R = M^e % N
  = 47^3 % 55
  = 103823 % 55
  = 38
```

I tested the decode program using: 38 27 55

```
The message to be decoded is 38.
The private (decoding) key is 27.
The N is 55.

The results are:

R = 38, d = 27, N = 55
Decoded Message Q = 47
```

This output is correct because:

```
Q = R^d % N
  = 38^27 % 55
  = ((38^3 % 55)^9) % 55
  = ((37)^9) % 55
  = ((37)^3 % 55)^3 % 55
  = (53)^3 % 55
  = 47
```

As you can see, the message Q decoded to the original message M.

Below are results for more, larger primes.

RESULTS -

Primes		Message	Create	Key Output		Encode Output M->R	Decode Output R->Q
p	q	M	e	N	d	R	Q
5	11	47	3	55	27	38	47
19	29	47	5	551	101	73	47
67	71	47	13	4757	1777	1132	47
151	157	5555	7	23707	3343	1780	5555
179	181	5555	7	32399	27463	9109	5555
223	227	5555	5	50621	20069	42643	5555
233	239	47	3	55687	36811	48136	47
239	241	47	11	57599	20771	29277	47
241	251	47	7	60491	17143	46592	47
241	89	5555	7	21449	18103	17521	5555