ELEC-270 Solutions to Assignment 3

1. Use the moduli 4, 9 and 35 to represent the integers 84 and 47. Using the modular representations of the integers, show the computations needed to get the product of 84 and 47.

Let $a = 84$ and $b = 47$
$m_1 = 4$, $m_2 = 9$, $m_3 = 35$

84 mod 4 = 0	47 mod 4 = 3
84 mod 9 = 3	47 mod 9 = 2
84 mod 35 = 14	47 mod 35 = 12

∴ (a)	$84 \equiv 0 \pmod 4$	$47 \equiv 3 \pmod 4$	
(b)	$84 \equiv 3 \pmod 3$	$47 \equiv 2 \pmod 9$	
(c)	$84 \equiv 14 \pmod{35}$	$47 \equiv 12 \pmod{35}$	
∴ (d)	$84 \cdot 47 \equiv 0 \cdot 3 \pmod 4$		by applying Theorem 5 on p. 242 to line (a)
(e)	$84 \cdot 47 \equiv 3 \cdot 2 \pmod 9$		ditto to (b)
(f)	$84 \cdot 47 \equiv 14 \cdot 12 \pmod{35}$		ditto to (c)

So, find x such that x solves the following set of congruences:

	$x \equiv 0 \pmod 4$	from (d)
	$x \equiv 6 \pmod 9$	from (e)
∴	$x \equiv 168 \pmod{35}$	from (f)
	$\equiv 28 \pmod{35}$	

Apply Chinese Remainder Theorem
where $a_1 = 0$, $a_2 = 6$, $a_3 = 28$
$m_1 = 4$, $m_2 = 9$, $m_3 = 35$
$M_1 = 9 \cdot 35 = 315$, $M_2 = 4 \cdot 35 = 140$, $M_3 = 4 \cdot 9 = 36$

Now, get inverses:
(1) get inverse of M_1 mod m_1 (315 mod 4)

$315 = 4 \cdot 78 + 3$
$4 = 3 \cdot 1 + 1$
$\therefore 1 = 4 - 3$
$= 4 - (315 - 4 \cdot 78)$
$= 79 \cdot 4 - 1 \cdot 315$
$\therefore -1$ is inverse of 315 mod 4, i.e., $-1 + 4 = 3$ is inverse too.

(2) get inverse of 140 mod 9

$140 = 9 \cdot 15 + 5$
$9 = 5 \cdot 1 + 4$
$5 = 4 \cdot 1 + 1$

$\therefore 1 = 5 - 4$
$= 5 - (9 - 5)$
$= 2 \cdot 5 - 9$
$= 2 \cdot (140 - 9 \cdot 15) - 9$
$= 2 \cdot 140 - 31 \cdot 9$

$\therefore 2$ is inverse of 140 mod 9

(3) get inverse of 36 mod 35
$36 = 35 \cdot 1 + 1$

$\therefore 1 = 36 - 35$
$= 1 \cdot 36 - 1 \cdot 35$

$\therefore 1$ is inverse of 36 mod 35

$\therefore x = a_1 M_1 y_1 + a_2 M_2 y_2 + a_3 M_3 y_3$
$= 0 + 6 \cdot 140 \cdot 2 + 28 \cdot 36 \cdot 1$
$= 2688$

Solution is congruent to 2688 mod $m_1 m_2 m_3$ where $m_1 \cdot m_2 \cdot m_3 = 4 \cdot 9 \cdot 35 = 1260$
$\therefore x \equiv 2688 \pmod{1260}$
$\equiv 168 \pmod{1260}$

Now we know that
$84 \cdot 47 = 168 + 1260k$ for some k
Without doing computation with large numbers, we know that $3200 < 84 \cdot 47 < 4500$ (By rounding 84 down to 80 and 47 down to 40 to get $8 \cdot 4 = 32$, i.e., lower bound is 3200, and similarly rounding 84 up to 90 and 47 up to 50 to get $9 \cdot 5 = 45$ and so upper bound is 4500.).
The only x in the range between 3200 and 4500 is
$x = 168 + 1260 \cdot 3$
$= 3948$
(which you can confirm is $84 \cdot 47$)
Note: if you'd used $a_3 = 168$ instead of 28, you'd still get the same final result.

2. **(a) Prove that for integers *a* and *m*, if *gcd(a,m)>1*, then there does not exist an inverse of *a* mod *m*.**
   (b) Explain why you couldn't use the RSA cryptosystem with primes *p* = 11, *q* = 19 and encryption exponent *e* = 5

   (a) We'll prove this by contradiction. Suppose that an inverse did exist, call it *d*. Then $a \cdot d \equiv 1 \pmod{m}$, by definition of inverse.
   $\Rightarrow a \cdot d \bmod m = 1 \bmod m$
   $= 1$
   $\Rightarrow a \cdot d = m \cdot k + 1$, for some integer *k*
   $\Rightarrow a \cdot d - m \cdot k = 1$, for some integer *k*
   Now, *gcd(a,m)* divides both *a* and *m* (by definition of greatest common divisor), so it divides $a \cdot d - m \cdot k$. From the above equation, since $a \cdot d - m \cdot k = 1$, this means that *gcd(a,m)* also divides 1. However, by assumption, *gcd(a,m)* is greater than 1, so it cannot possibly divide 1 (i.e., a number greater than 1 cannot divide 1). This leads to a contradiction!

   Note: The following is **not** an adequate proof:
   "*gcd(a,m)>1*, i.e., *gcd(a,m)* $\neq 1$, which means that *a* and *m* are not relatively prime. Therefore we know that no inverse exists."

   What we proved in class is that **if** *gcd*(*a*,*b*) =1, **then** there is an inverse of *a* mod *b*. We did not prove the converse, namely that if *gcd*(*a*,*b*) $\neq 1$, then there is no inverse of *a* mod *b*.

   (b) For RSA to work, you need to find an inverse of *e* mod (*p*-1)(*q*-1).
   $(p-1)(q-1) = 10 \cdot 18 = 180$
   $e = 5$
   From (a), we know that no such inverse exists (since gcd(5,180)=5).

3. **(a) Detail how you would encrypt the message MATH using the RSA system with prime numbers *p* = 43 and *q* = 59 and exponent *e* = 5. You do not need to perform the modular exponentiation; just list the computations that would need to be performed.**
   (b) Explain how each modular exponentiation in (a) can be done using only 3 multiplications on a basic calculator.
   (c) Show why the exponent given in (a) is a valid exponent to use for this encryption.

(a)

A	00	N	13
B	01	O	14
C	02	P	15
D	03	Q	16
E	04	R	17
F	05	S	18
G	06	T	19
H	07	U	20
I	08	V	21
J	09	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

M	A	T	H
12	00	19	07

$p \cdot q = 43 \cdot 59$
$= 2537$
$e = 5$

Encrypt as
$C_1 = 1200^5 \bmod 2537$
$C_2 = 1907^5 \bmod 2537$

(b) $1200^2 \bmod 2537 = 1521$
$1521^2 \bmod 2537 = 2234$
$2234 \cdot 1200 \bmod 2537 = 1728$

$1907^2 \bmod 2537 = 1128$
$1128^2 \bmod 2537 = 1347$
$1347 \cdot 1907 \bmod 2537 = 1285$

(c) Must check if e is relatively prime to $(p-1)(q-1) = 42 \cdot 58 = 2438$
$2436 = 487 \cdot 5 + 1$
$5 = 5 \cdot 1 + 0$
$\therefore gcd(2436, 5) = 1$ since last nonzero remainder is 1.

4. (a) Write and test a computer program that implements Algorithm 5 in Section 4.2 (6th edition: Algorithm 5 in Section 3.6), which is an algorithm for performing modular exponentiation.
(b) Encrypt the message ATTACK using the RSA system with $n = 43 \cdot 59$ and $e = 13$, translating each letter into integers and grouping together pairs of integers as done in class. You will need to use the computer program from (a) to perform fast modular exponentiation.

(c) What is the original message encrypted using RSA with $n = 53 \cdot 61$ and $e = 17$ if the encrypted message is 3185 2038 2460 2550?

(a)

Program in C

```
#include<stdio.h>

/* Program to do fast modular exponenentiation
 adapted from program written by Kim Edwards */

long modularExp(long, long, long);

/* main program just gets user to input the number b, the
 exponent n and the modulus m */
int main ()
{
     long b,n,m, result;
     printf ("b=");
     scanf("%d",&b);
     printf ("n=");
     scanf("%d",&n);
     printf ("m=");
     scanf("%d",&m);
     result=modularExp(b,n,m);
     printf("\n%d\n\n",result);
     return 0;
}

/* The modularExp()function uses the squaring and multiply
   algorithm to increase the computation speed.
   It outputs b^n mod m.*/

long modularExp(long b, long n, long m)
{
     long power, answer;

     power=b%m;   // do b mod m
     answer = 1;
     while (n>0)
     {
          if ((n%2)==1)     // checks if bit=1; binary expansion
               answer = (answer*power)%m;
          n=n>>1;           // shifts bits right once to get next bit
          if (n>0)          // avoids unnecessary last calculation
               power=(power*power)%m;    // successive squaring then
                                         // mod m
     }
     return answer;
}
```

Program in Java

```
import java.util.*;
import java.io.*;

public class DiscMath {
   public static void sqrmult(int X, int N,int  M){
      int z, answer;
      z = X%M;
      answer = 1;

      while (N>0){
         if ((N%2)==1){
            answer = (answer*z)%M;
         }
         N = N>>1;
         if (N>0){
            z = (z*z) %M;
         }
      }
      System.out.println("Answer : " + answer);
   }
   public static void main (String args[]){
      BufferedReader stdin =
         new BufferedReader(new InputStreamReader(System.in));
      String digitstring = "";
      int X,M,N;
      System.out.println("x = ");
      try{
         digitstring =stdin.readLine();
      }
      catch ( Exception e ) {
         System.err.println( " Error");
      }
      X = Integer.parseInt(digitstring);
      System.out.println("n = ");
      try{
         digitstring = stdin.readLine();
      }
      catch ( Exception e ) {
         System.err.println( " Error");
      }
      N = Integer.parseInt(digitstring);
```

```
      System.out.println("m = ");
      try{
         digitstring = stdin.readLine();
      }
      catch ( Exception e ) {
         System.err.println( " Error");
      }
      M = Integer.parseInt(digitstring);
      sqrmult(X, N, M);
   }
}
```

(b) Translating the letters into numbers we have 0019 1900 2100.
Thus we need to compute $C = P^{13} \bmod 2537$ for $P = 19$, $P = 1900$, and $P = 210$. The results of these calculations, done by fast modular exponentiation on a computer are 2299, 1317, and 2117, respectively. Thus the encrypted message is 2299 1317 2117.

(c) First we have to find the decryption key d, which is an inverse of 17 modulo $52 \cdot 60$, i.e., an inverse of 17 mod 3120.

To get an inverse, we do the Euclidean Algorithm:
$3120 = 17 \cdot 183 + 9$
$17 = 9 \cdot 1 + 8$
$9 = 8 \cdot 1 + 1$

Working backwards and backsubstituting:
$1 = 9 - 8$
$= 9 - (17 - 9 \cdot 1)$
$= 2 \cdot 9 - 17$
$= 2 \cdot (3120 - 17 \cdot 183) - 17$
$= 2 \cdot 3120 - 367 \cdot 17$
So -367 is an inverse. In RSA, the convention is that the inverse is positive. We can add multiples of the modulus to the inverse and the new value will also be an inverse. So we pick -367+ 3120, which equals 2753, for our inverse.

We have d=2753. Recall that $n = 53 \cdot 61 = 3233$. Now we need to compute $C^{2753} \bmod 3233$ for each of the four given blocks of cyphertext.

Using our modular exponentiation program, we get
$3185^{2753} \bmod 3233 = 1816$
$2038^{2753} \bmod 3233 = 2008$
$2460^{2753} \bmod 3233 = 1717$
$2550^{2753} \bmod 3233 = 0411$

Now we translate back into letters:
18 16 20 08 17 17 04 11
S Q U I R R E L