```
RSA program Report / Started running on: Thu Nov 26 21:58:55 2009

****************************************************************

****************************************************************
****************************************************************
****************************************************************




 TRACE OF GENERATING SAMPLE INTEGER
 ==================================
 random number  : 0 is: 25024
 In binary it is:       110000111000000
 The least significant bit of that number is obviously ==> 0
 So therefore bit number 1  from left is:                  0

 random number  : 1 is: 28864
 In binary it is:       111000011000000
 The least significant bit of that number is obviously ==> 0
 So therefore bit number 2  from left is:                  0

 random number  : 2 is: 25294
 In binary it is:       110001011001110
 The least significant bit of that number is obviously ==> 0
 So therefore bit number 3  from left is:                  0

 random number  : 3 is: 1535
 In binary it is:       10111111111
 The least significant bit of that number is obviously ==> 1
 So therefore bit number 4  from left is:                  1

 random number  : 4 is: 4316
 In binary it is:       1000011011100
 The least significant bit of that number is obviously ==> 0
 So therefore bit number 5  from left is:                  0



 32-bit Padded random integer: 00000000000000000000000001000101
```

THIS IS A SAMPLE FOR AN UNSUCCESSFUL PRIMALITY CHECK
==========

Checking the integer: 115 , Using a= 58

i	X[i]	z	y	y
6	1	1	1	58
5	1	58	29	72
4	1	72	9	62
3	0	62	49	49
2	0	49	101	101
1	1	101	81	98
0	0	98	59	59

The integer: 115 is not a prime!

THIS IS A SAMPLE FOR A SUCCESSFUL PRIMALITY CHECK (PERHAPS PRIME)
==========

Checking the integer: 101 , Using a= 79

i	X[i]	z	y	y
6	1	1	1	79
5	1	79	80	58
4	0	58	31	31
3	0	31	52	52
2	1	52	78	1
1	0	1	1	1
0	0	1	1	1

The integer: 101 is perhaps a prime!

Our n=8989 Our Phi(8989)=8800

Checking if e=5 can be a public key [Extended Euclidean Algorithm]

q	r1	r2	r	s1	s2	s	t1	t2	t
1760	8800	5	0	1	0	1	0	1	-1760

Unfortunately e=5 Does not have a multiplicative inverse in Z(8800), so we go ahead and try the next e

Checking if e=6 can be a public key [Extended Euclidean Algorithm]

q	r1	r2	r	s1	s2	s	t1	t2	t
1466	8800	6	4	1	0	1	0	1	-1466
1	6	4	2	0	1	-1	1	-1466	1467
2	4	2	0	1	-1	3	-1466	1467	-4400

Unfortunately e=6 Does not have a multiplicative inverse in Z(8800), so we go ahead and try the next e

Checking if e=7 can be a public key [Extended Euclidean Algorithm]

q	r1	r2	r	s1	s2	s	t1	t2	t
1257	8800	7	1	1	0	1	0	1	-1257
7	7	1	0	0	1	-7	1	-1257	8800

FOUND IT ! e will be:7 Its multiplicative inverse in Z(8800) is:7543
So, ==> d=7543

HERE IS Alice'S CRYPTO SYSTEM

As Integers:

```
              p = 101
              q = 89
              n = 8989
              e = 7
              d = 7543

 As BIT strings:
              p = 1100101
              q = 1011001
              n = 10001100011101
              e = 111
              d = 1110101110111


 Here are: r, h(r) and s as bits
 -------------------------------

 r
=00100000010000010110110001101001011000110110010100000000000000000001000110001110
      10000000000000000000000000000111
 h(r)=00000000000000000000000001011011
 s   =00000000000000000000010000000100


 Here are: h(r) and s as integers
 --------------------------------
 h(r)=91
 s   =1028




NOT REQUIRED PORTION
#####################################################################
=======> Trent's keys: Public => (11, 14351)  Private => (1283, 14351)
#####################################################################
END OF NOT REQUIRED PORTION



 So here is Alice's certificate
================================
 Alice_cert.name=  Alice
 Alice_cert.e   = 7
 Alice_cert.r   =
00100000010000010110110001101001011000110110010100000000000000000001000110001110100
                  00000000000000000000000000000111
 Alice_cert.s   = 1028
```

Alice's 'n' in 32 bits is:00000000000000000010001100011101

K ===> Bob found first '1' from left of 'n' in position ===> K= 13

So the u composed by Bob (according to the specs):

u[b]=00000000000000000001011110011111
This u as integer is ============================> u =6047

So Bob sends to Alice u= 6047

And then Alice [after hashing and decrypting with her private key] returns to Bob v= 603

Showing Bob's encryption of V E(V, e)
==========

i	X[i]	Squaring Y	Multiplying Y
3	1	1^2(mod 8989)= 1	603x 1(mod 8989)= 603
2	1	603^2(mod 8989)= 4049	603x 4049(mod 8989)= 5528
1	1	5528^2(mod 8989)= 5173	603x 5173(mod 8989)= 136

Bob encrypts v with Alice's public key and gets Z= 136
Bob also does hash of: 6047 which in binary is:
00000000000000000001011110011111 and he gets h(u)= 136

AND IF THEY MATCH [Z == h(u)], BOB KNOWS HE IS TALKING TO SOMEONE WHO'S GOT ALICE'S PRIVATE KEY (HOPEFULLY ALICE)

TO SUMMARIZE

In Integers:

```
u           = 6047
h(u)        = 136
V=D(d,h(u)) = 603
E(e,V)      = 136
```

In Binary:

```
u           = 00000000000000000001011110011111
h(u)        = 00000000000000000000000010001000
V=D(d,h(u)) = 00000000000000000000001001011011
E(e,V)      = 00000000000000000000000010001000
```