Pimpri Chinchwad College of Engineering
Computer Laboratory - III

Assignment No - 5

1 Aim

Write a Python/ Java/c++ program to validate the parameter tuple {p,q,g} for the security of the DSA. Use Miller-Rabin primality test determine whether a given number is composite or probably prime.

2 Objective

- To Understand how to do Primality Test.
- To learn Miller Rabin Algorithm for Primality Test.

3 Software Requirements

- Windows
- Java

4 Mathematical Model

S = s, e, x, y, Fme, DD, NDD

S = Initial State

E = End State
X = Input Value
X={x1}
where, x1= p

Y = Output
Y={y1}
y1={True/False}
Fme = Main function
Fme={f1}
f1="Miller Rabin algorithm is used to check primality of a given number".

DD = Deterministic data {Number p}
NDD = Non Deterministic Data {Primality }

5 Theory

5.1 Primality Test

- Given an integer n, how can we tell if n is prime? The most obvious way is to look for factors of n, but no efficient factoring algorithm is known.
- From now let us assume n is odd, since deciding the primality of an even number is trivial.
- By Fermats Theorem, if **n** is prime, then for any **a** we have $a^{n-1} = 1(mod n)$. This suggests the Fermat test for a prime: pick a random a epsilon{1,...,n-1} and see if $a^{n-1} = 1(mod n)$. If not, then n must be composite.

5.2 The Miller Rabin Test

- A primality test that provides an efficient probabilistic algorithm for determining if a given number is prime. It is based on the properties of strong pseudoprimes.

- The algorithm proceeds as follows. Given an odd integer n, let $n = 2^r s + 1$ with s odd. Then choose a random integer a with $1 <= a <= n-1$. If $a^s = 1(modn)$ or $a^{(2^j s)} = -1(modn)$ for some $0 <= j <= r-1$, then n passes the test. A prime will pass the test for all a.

- The test is very fast and requires no more than (1+o(1))logn multiplications (mod n), where log is the logarithm base 2. Unfortunately, a number which passes the test is not necessarily prime. Monier (1980) and Rabin (1980) have shown that a composite number passes the test for at most 1/4 of the possible bases a. If N multiple independent tests are performed on a composite number, then the probability that it passes each test is $(1/4)^N$or less.

- However, if the smallest composite number that passes a particular set of tests is known ahead of time, then that set of tests constitutes a primality proof for all smaller numbers. The sequence of smallest odd numbers passing a multiple Rabin-Miller test using the first k primes for k=1, 2, ... is given by 2047, 1373653, 25326001, 3215031751, 2152302898747, 3474749660383, 341550071728321, 341550071728321, ... (OEIS A014233; Jaeschke 1993). Therefore, multiple Rabin tests using the first 7 primes (using 8 gives no improvement) are valid for every number up to $3.4 * 10^{(14)}$.

- The Wolfram Language implements the multiple Rabin-Miller test in bases 2 and 3 combined with a Lucas pseudoprime test as the primality test used by the function Prime Q[n]. As of 1997, this procedure is known to be correct only for all $n < 10^{(16)}$, but no counterexamples are known and if any exist, they are expected to occur with extremely small probability (i.e., much less than the probability of a hardware error on a computer performing the test)

6 Algorithm

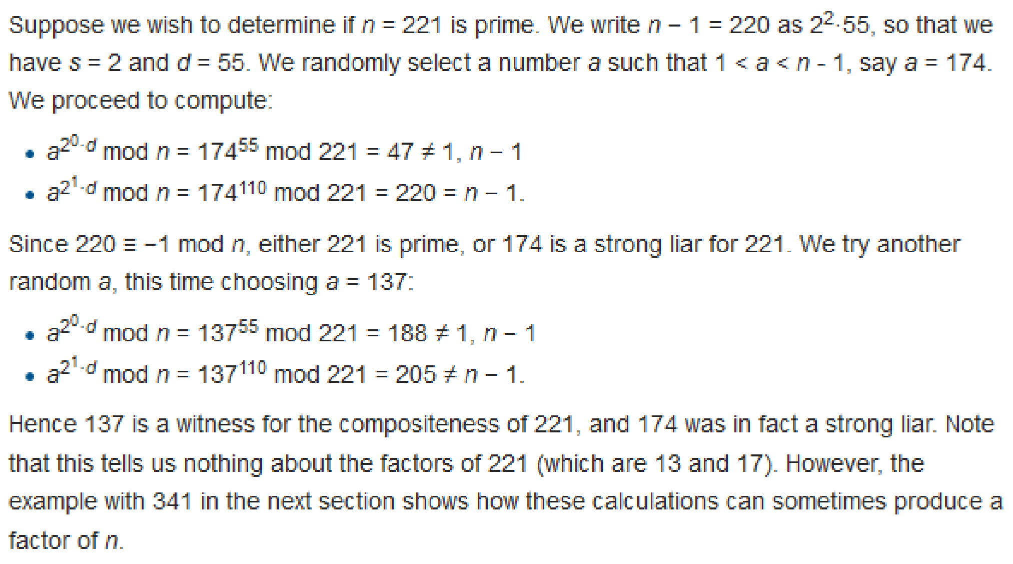
```
Input: n > 2, an odd integer to be tested for primality;
       k, a parameter that determines the accuracy of the test
Output: composite if n is composite, otherwise probably prime
write n - 1 as 2s.d with d odd by factoring powers of 2 from n - 1
LOOP: repeat k times:
    pick a randomly in the range [2, n - 1]
   x  ad mod n
   if x = 1 or x = n - 1 then do next LOOP
   for r = 1 .. s - 1
      x  x2 mod n
      if x = 1 then return composite
      if x = n - 1 then do next LOOP
   return composite
return probably prime
```

6.1 Parameter Generation

The first part of the DSA algorithm is the public key and private key generation, which can be described as:

1. Choose a prime number q, which is called the prime divisor.
2. Choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus.
3. Choose an integer g, such that 1 < g < p, g^q mod p = 1 and g = $h^{(p1)/q)} mod p$. q is also called g's multiplicative order modulo p.
4. Choose an integer, such that 0 < x < q. Compute y as g^x mod p.
5. Package the public key as p,q,g,y. Package the private key as p,q,g,x.

7 Example



Suppose we wish to determine if $n = 221$ is prime. We write $n - 1 = 220$ as $2^2 \cdot 55$, so that we have $s = 2$ and $d = 55$. We randomly select a number a such that $1 < a < n - 1$, say $a = 174$. We proceed to compute:

- $a^{2^0 \cdot d} \bmod n = 174^{55} \bmod 221 = 47 \neq 1, n - 1$
- $a^{2^1 \cdot d} \bmod n = 174^{110} \bmod 221 = 220 = n - 1$.

Since $220 \equiv -1 \bmod n$, either 221 is prime, or 174 is a strong liar for 221. We try another random a, this time choosing $a = 137$:

- $a^{2^0 \cdot d} \bmod n = 137^{55} \bmod 221 = 188 \neq 1, n - 1$
- $a^{2^1 \cdot d} \bmod n = 137^{110} \bmod 221 = 205 \neq n - 1$.

Hence 137 is a witness for the compositeness of 221, and 174 was in fact a strong liar. Note that this tells us nothing about the factors of 221 (which are 13 and 17). However, the example with 341 in the next section shows how these calculations can sometimes produce a factor of n.

8 Testing

8.1 Positive Testing

Input: Prime Number.
Output:Number is Composite.

8.2 Negative Testing

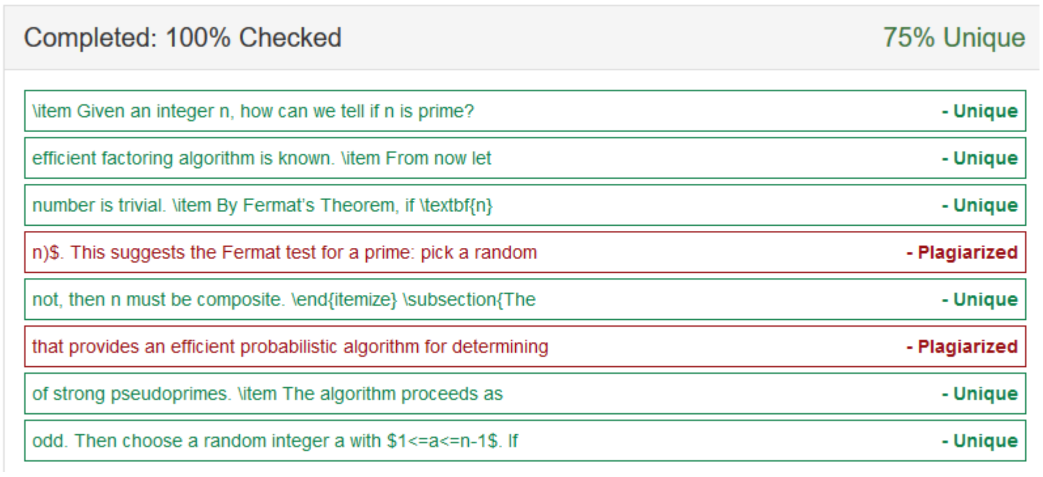
Input:Non Prime Number.
Output:Number is Inconclusive.

9 Conclusion

Thus, We have studied and implemented miller-rabin algorithm to perform primality test.

Roll No.	Name of Student	Date of Performance	Date of Submission
302	Abhinav Bakshi	18/02/2016	10/03/2016

10 Plagarism Report


Completed: 100% Checked | 75% Unique

\item Given an integer n, how can we tell if n is prime?	- Unique
efficient factoring algorithm is known. \item From now let	- Unique
number is trivial. \item By Fermat's Theorem, if \textbf{n}	- Unique
n)$. This suggests the Fermat test for a prime: pick a random	- Plagiarized
not, then n must be composite. \end{itemize} \subsection{The	- Unique
that provides an efficient probabilistic algorithm for determining	- Plagiarized
of strong pseudoprimes. \item The algorithm proceeds as	- Unique
odd. Then choose a random integer a with $1<=a<=n-1$. If	- Unique

11 Output

```
---------------------------------------------
Miller Rabin output

Enter Number to test :
71
Inconclusive


Generation
Parameters :
p : 2983
q : 71
g : 425

--------------------------------------------
```