Assignment No.

Title: Primality Testing using Miller Rabin method.

Aim: Write a Python/ Java program to validate the parameter tuple for the security of the DSA. Design necessary classes. Use Miller-Rabin primality testing may be used.

Objective:

- To understand the basics of primality testing.

Theory:

Primality testing is an important algorithmic problem. In addition to being a fundamental mathematical question, the problem of how to determine whether a given number is prime has tremendous practical importance. For many cryptographic algorithms, it is necessary to select one or more very large prime numbers at random. As an example, every time the RSA public-key cryptosystem is used, it is needed to generate a private key consisting of two large prime numbers and a public key consisting of their product. To do this, one needs to be able to check rapidly whether a number is prime .

In 1980, Michael Rabin discovered a randomized polynomial-time algorithm to test whether a number is prime. It is called the Miller-Rabin primality test because it is closely related to a deterministic algorithm studied by Gary Miller in 1976. This is still the most practical known primality testing algorithm, and is widely used in software libraries that rely on RSA encryption, e.g. OpenSSL

The algorithm due to Miller and Rabin is typically used to test a large number for primality. Before explaining the algorithm, we need some background. First, any positive odd integer

$n \geq 3$ can be expressed as follows:

$n - 1 = 2^k q$ with $k > 0$, q odd and n - 1 is an even integer. Then, divide (n- 1) by 2 until the result is an odd number q, for a total of k divisions. If n is expressed as a binary number, then the result is achieved by shifting the number to the right until the rightmost digit is a 1, for a total of k shifts.

Two properties of prime numbers:

The first property is stated as follows: If p is prime and a is a positive integer less than p, then a^2 mod p = 1 if and only if either a mod p = 1 or a mod p= 1 mode p = p-1. By the rules of modular arithmetic (a mode p) (a mode p) = a^2 mod p. Conversely, if a^2 mod p = 1, then $(a \bmod p)^2 = 1$, which is true only for a mod p = 1

The second property is stated as follows: Let p be a prime number greater than 2. Then

$p - 1 = 2^k q$, with k > 0 q is odd

The procedure TEST takes a candidate integer n as input and returns the result composite if n is definitely not a prime, and the result inconclusive if n may or may not be a prime.

```
TEST (n)
1.  Find integers k, q, with k > 0, q odd, so that $(n - 1 = (2^k q)$;
2.  Select a random integer a, $1 < a < n-1$;
3.  if $a^q$ mod n = 1 then return ("inconclusive");
4.  for j = 0 to k - 1 do
5.      if $a^{2jq}$ mod n = n - 1 then return ("inconclusive");
6.  return ("composite");
```

Repeated Use of the Miller-Rabin Algorithm:

Given an odd number n that is not prime and a randomly chosen integer, a with $1 < a < n-1$, the probability that TEST will return inconclusive (i.e., fail to detect that n is not prime) is less than 1/4. Thus, if t different values of a are chosen, the probability that all of them will pass TEST (return inconclusive) for n is less than (1/4)^t For example, for t = 10, the probability that a nonprime number will pass all ten tests is less than 106. Thus, for a sufficiently large value of t, we can be confident that n is prime if Miller's test always returns inconclusive

This gives us a basis for determining whether an odd integer n is prime with a reasonable degree of confidence. The procedure is as follows: Repeatedly invoke TEST (n) using randomly chosen values for a. If, at any point, TEST returns composite, then n is determined to be nonprime. If TEST continues to return inconclusive for t tests, for a sufficiently large value of t, assume that n is prime

Conclusion: Thus we have successfully implemented the Miller Rabin primality test and verified the same for large numbers.