**Lab 3**

**Lab 3.1: Write a program to implement the Miller-Rabin primality test. Test it with various values of 'n'.**

**Algorithm:**

bool isPrime(int n, int k)

1. Handle base cases for n < 3
2. If n is even, return false.
3. Find an odd number d such that n-1 can be written as d\*2r. Note that since n is odd, (n-1) must be even and r must be greater than 0.
4. Do following k times
   * if (millerTest(n, d) == false)
   * return false
5. Return true.

**Source Code:**

#include <iostream>

#include <stdlib.h>

using namespace std;

long long mulmod(long long, long long, long long);

long long modulo(long long, long long, long long);

bool Miller(long long, int);

int main()

{

// generateHeader("Program to implement Miller Rabin primality test");

do

{

int iteration = 10;

long long num;

cout << "Enter integer to test primality: ";

cin >> num;

if (Miller(num, iteration))

cout << num << " is prime" << endl;

else

cout << num << " is not prime" << endl;

char choice;

cout << "Do you want to continue? (y/n): ";

cin >> choice;

if (choice == 'n' || choice == 'N')

break;

} while (true);

cin.get();

return 0;

}

long long mulmod(long long a, long long b, long long m)

{

long long x = 0,

y = a % m;

while (b > 0)

{

if (b % 2 == 1)

{

x = (x + y) % m;

}

y = (y \* 2) % m;

b /= 2;

}

return x % m;

}

long long modulo(long long base, long long e, long long m)

{

long long x = 1;

long long y = base;

while (e > 0)

{

if (e % 2 == 1)

x = (x \* y) % m;

y = (y \* y) % m;

e = e / 2;

}

return x % m;

}

bool Miller(long long p, int iteration)

{

if (p < 2)

{

return false;

}

if (p != 2 && p % 2 == 0)

{

return false;

}

long long s = p - 1;

while (s % 2 == 0)

{

s /= 2; }

for (int i = 0; i < iteration; i++)

{

long long a = rand() % (p - 1) + 1, temp = s;

long long mod = modulo(a, temp, p);

while (temp != p - 1 && mod != 1 && mod != p - 1)

{

mod = mulmod(mod, mod, p);

temp \*= 2;

}

if (mod != p - 1 && temp % 2 == 0)

{

return false;

}

}

return true;

}