Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

# ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Системи безпеки програм і даних» на тему
«Розробка програми генератора великих простих чисел»

ВИКОНАВ:
студент III курсу ФІОТ
групи ІВ-91
Красновський Олексій

ПЕРЕВІРИВ:
аспірант
Іваніщев Б.В.

Київ – 2021

**Завдання:**

Розробка програми генератора великих простих чисел (ВПЧ). Для шифрування і розрахунку ключів за схемою RSA необхідно використовувати два великих простих десяткових числа з кількома десятками десяткових розрядів (за варіантами). Для генерації таких простих чисел можна використовувати формули у відповідності з тестом Рабіна або іншими алгоритмами, але для перевірки властивостей сформованих кандидатів у прості числа необхідно використовувати малу теорему Ферма і алгоритм ШДП.

**Аналіз засобів рішення задачі:**

При аналізі завдання лабораторної роботи 4, проблем не виникло. Єдиним недоліком вважаю використання стандартної імплементації BigInteger та BigDecimal стандартної бібліотеки Java, а не написаної власної.

**Лістинг програми:**

**RabinPrimalityTestUtils.java**

```
package util;


import java.math.BigInteger;
import java.util.Random;


public class RabinPrimalityTestUtils {


  private static final int[] smallPrimeNumbers = {2, 3, 5, 7, 11, 13};


  public static BigInteger getPrimeNumber(long PMax) {
    BigInteger a = BigInteger.valueOf(getRandomSmallPrimeNumber());
    BigInteger K = BigInteger.valueOf(Math.round(customLog(a.doubleValue(), (double) PMax / 2)));
    BigInteger P1 = pow(a, K).multiply(BigInteger.TWO).add(BigInteger.ONE);
    BigInteger P2 = pow(a, K).multiply(BigInteger.TWO).subtract(BigInteger.ONE);
    while (true) {
      if (isPrime(a, P1)) return P1;
```

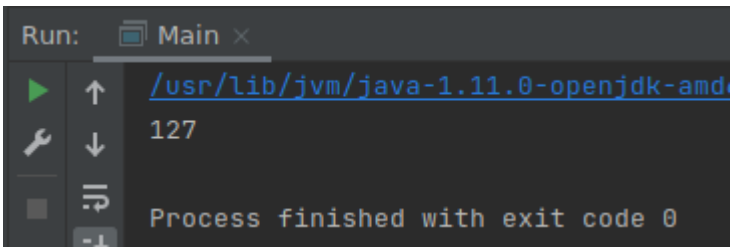
```
      if (isPrime(a, P2)) return P2;
      P1 = P1.add(BigInteger.TWO);
      P2 = P2.subtract(BigInteger.TWO);
    }
  }

  private static int getRandomSmallPrimeNumber() {
    return smallPrimeNumbers[new Random().nextInt(smallPrimeNumbers.length)];
  }

  public static double customLog(double base, double logNumber) {
    return Math.log(logNumber) / Math.log(base);
  }

  private static boolean isPrime(BigInteger a, BigInteger P) {
    BigInteger powResult = pow(a, P.subtract(BigInteger.ONE));
    BigInteger result = powResult.mod(P);
    return result.equals(BigInteger.ONE);
  }

  public static BigInteger pow(BigInteger base, BigInteger exponent) {
    BigInteger result = BigInteger.ONE;
    while (exponent.signum() > 0) {
      if (exponent.testBit(0)) result = result.multiply(base);
      base = base.multiply(base);
      exponent = exponent.shiftRight(1);
    }
    return result;
  }
}
```

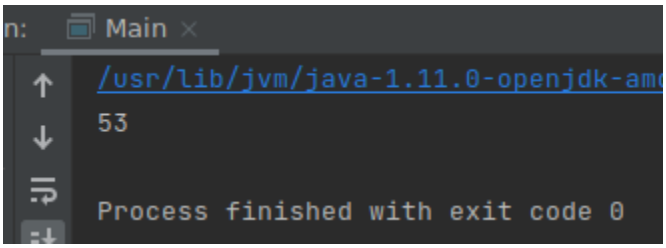
## Main.java

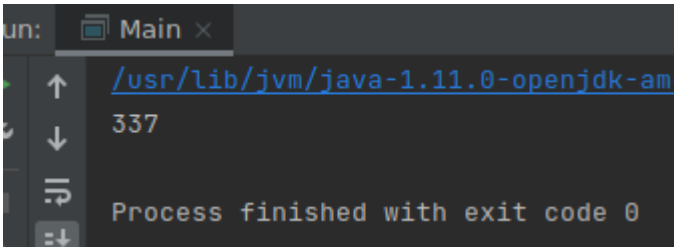
```
import static util.RabinPrimalityTestUtils.pow;

import java.math.BigInteger;
import util.RSAUtils;
import util.RabinPrimalityTestUtils;
```

```
public class Main {


  public static void main(String[] args) {
    // lab 4
    System.out.println(RabinPrimalityTestUtils.getPrimeNumber(100));
  }
}
```

**Скріншоти виконання:**




**Висновки:**

При виконанні даної лабораторної роботи проблем не виникло так як матеріал лекцій чудово покриває її тему.

Зважаючи на це, завдання лабораторної роботи виконано. Робота програми демонструє правильність обраних рішень.