import java.math.BigInteger;

import java.security.SecureRandom;

class RSADemo{

    public static final BigInteger ONE = BigInteger.ONE;

    public static final SecureRandom RANDOM = new SecureRandom();

    public final BigInteger privateKey;

    private final BigInteger publicKey;

    private final BigInteger modulus;

    public RSADemo(int bitLength) {

        BigInteger p = BigInteger.probablePrime(bitLength / 2, RANDOM);

        BigInteger q = BigInteger.probablePrime(bitLength / 2, RANDOM);

        BigInteger phi = (p.subtract(ONE)).multiply(q.subtract(ONE));

        modulus = p.multiply(q);

        publicKey = BigInteger.valueOf(65537);

        privateKey = publicKey.modInverse(phi);

        System.out.println("Prime p: " + p);

        System.out.println("Prime q: " + q);

        System.out.println("Modulus: " + modulus);

        System.out.println("Euler's Totient(φ): " + phi);

    }

    public BigInteger encrypt(BigInteger message) {

        return message.modPow(publicKey, modulus);

    }

    public BigInteger decrypt(BigInteger encrypted) {

        return encrypted.modPow(privateKey, modulus);

    }

    @Override

    public String toString() {

        return "Public key: " + publicKey + "\n" +

               "Private key: " + privateKey + "\n" +

               "Modulus: " + modulus;

    }

    public static void main(String[] args) {

        if (args.length == 0) {

            System.out.println("Usage: java RSADemo <bit-length>");

            return;

        }

        int bitLength = Integer.parseInt(args[0]);

        RSADemo rsa = new RSADemo(bitLength);

        System.out.println(rsa);

        BigInteger message = BigInteger.valueOf(8);

        BigInteger encrypted = rsa.encrypt(message);

        BigInteger decrypted = rsa.decrypt(encrypted);

        System.out.println("Original message: " + message);

        System.out.println("Encrypted message: " + encrypted);

        System.out.println("Decrypted message: " + decrypted);

    }

}