```
// To implement Rabin Cryptosystem

import java.math.*;
import java.util.*;

class prac6
{
    static int plaintext = 1, n = 1;
    static int p = 1, q = 1, ciphertext = 1;

    static int RabinEncryption()
    {
        int sq = 1;
        //finding P^2
        for (int i = 1; i <= 2; i++)
        {
            sq = sq * plaintext;
        }

        // C = P^2 mod n
        ciphertext = sq % n;
        return ciphertext;
    }

    static void RabinDecryption()
    {
        // calculate a1, a2, b1, b2

            // find a1 and a2
            //a1 = +(C^((p+1)/4 mod p)
            int power = (p + 1)/4;
            BigInteger result = new BigInteger("1");
            // finding m to the power e
            for (int i = 1; i <= power; i++)
            {
                result = result.multiply(BigInteger.valueOf(ciphertext));
            }
            BigInteger a1 = result.mod(BigInteger.valueOf(p));
            BigInteger a2 = BigInteger.valueOf(p).subtract(a1);
        System.out.print("\na1 = "+a1);
        System.out.print(", a2 = "+a2);

        // find b1 and b2
        //b1 = +(C^((q+1)/4 mod q)
        int power2 = (q + 1)/4;
        BigInteger result2 = new BigInteger("1");
        // finding m to the power e
        for (int i = 1; i <= power2; i++)
        {
            result2 = result2.multiply(BigInteger.valueOf(ciphertext));
        }
        BigInteger b1 = result2.mod(BigInteger.valueOf(q));
        BigInteger b2 = BigInteger.valueOf(q).subtract(b1);

    System.out.print("\nb1 = "+b1);
    System.out.print(", b2 = "+b2);

    // Biginteger to integer
    int a1_i = a1.intValue();
    int a2_i = a2.intValue();
    int b1_i = b1.intValue();
    int b2_i = b2.intValue();

    System.out.print("\n\n");
    System.out.println("Four possible plaintext values:: "
    +ChineseRemainderTheorem(a1_i,b1_i,p,q) + ", "
    + ChineseRemainderTheorem(a1_i,b2_i,p,q) + ", "
    + ChineseRemainderTheorem(a2_i,b1_i,p,q) + ", "
    + ChineseRemainderTheorem(a2_i,b2_i,p,q));
    System.out.print("\n");
}
```

```
static int ChineseRemainderTheorem(int a1, int b1, int p, int q)
{
    int M = p * q;
    int M1 = M / p;
    int M2 = M / q;

     //finding multiplicative inverse
     int flag = 1, min =1;
     while (flag == 1)
     {
         for (int i=1; i<p; i++)
         {
             if (((i*M1)%p)==1)
             {
                 min = i;
                 flag = 0;
             }
         }
     }
    int M1_inv = min;

    //finding multiplicative inverse
     while (flag == 1)
     {
         for (int i=1; i<q; i++)
         {
             if (((i*M2)%q)==1)
             {
                 min = i;
                 flag = 0;
             }
         }
     }
    int M2_inv = min;

    int x = (a1*M1*M1_inv + b1*M2*M2_inv) % M;

    return x;
}
```

```
static int checkPrime(int n)
{
    // checking prime or not
    Scanner scan2 = new Scanner(System.in);

    boolean prime = false;
    boolean expr = false;
    while(prime == false)
    {
        if(n%6==1 || n%6==5 || n==2 || n==3)
        {
            prime = true;
        }
        else
        {
            System.out.print("Number is not prime. Enter again:: ");
            n = scan2.nextInt();
        }
    }

    //check if prime number is in the form 4k + 3
    while(expr==false)
    {
        for(int k = 0; k < 15; k++)
        {
            if( n == (4*k + 3))
            {
                expr = true;
            }
        }
        if(expr == false)
        {
            System.out.print("Number is not in the form '4K + 3'.
            Enter again:: ");
            n = scan2.nextInt();
        }
    }
    return n;

}
```

```
    public static void main(String args[])
    {
        Scanner scan = new Scanner(System.in);
        // input prime numbers and store in p and q
        System.out.print("\nEnter two prime numbers p and q.
                         \n(in the form 4k + 3)");
        System.out.print("\np = ");
        p = scan.nextInt();
        p = checkPrime(p);
        System.out.print("q = ");
        q = scan.nextInt();
        q = checkPrime(q);

        // public key
        n = p*q;

        // input plaintext
        System.out.print("\nPlaintext = ");
        plaintext = scan.nextInt();

        // encryption
        System.out.print("\nAfter encryption.\nCiphertext = “
        + RabinEncryption());

        // decryption
        System.out.print("\n\nAfter decryption.");
        RabinDecryption();
    }
}
```

OUTPUT:

```
Viddeshs-MacBook-Air:prac6 viddeshgharpure$ java prac6

Enter two prime numbers p and q.
(in the form 4k + 3)
p = 23
q = 7

Plaintext = 24

After encryption.
Ciphertext = 93

After decryption.
a1 = 1, a2 = 22
b1 = 4, b2 = 3

Four possible plaintext values:: 24, 116, 45, 137
```