ACM Pacific NW Region Programming Contest
11 November 2000

PROBLEM A
The Quadratic formula (Mod p)

The Quadratic Equation was a topic that preoccupied you for some time in Algebra. In this problem you'll be revisiting this topic. You are to find the roots of a quadratic equation:

$$a\ x^2 + b\ x + c \equiv 0 \pmod{p}$$

with a little twist. The coefficients **a**, **b** and **c** as well as **x** are positive integers in the range **1...p** where **p** is an odd prime number (that is included as part of the input). All operations are done **(Mod p)**. For instance, consider the equation **3 x^2 + 1000 x + 65709 ≡ 0 (Mod p)** and **p = 337639**. (You may trust us, **337639** is a prime number). You may verify **x = 2345** is one root of this quadratic **(Mod p)**.

One way to solve a modular quadratic is to use the good old Quadratic formula. The only caveat is how to perform the operations needed in the quadratic formula (**efficiently!**). For example, we need:

the **"power" (Mod p)** operation,
the **"square root" (Mod p)** operation, and finally
the **"division" (Mod p)** operation.

<u>Modular Power Operation</u>

Modular power is defined by the equation: **$(a)^b$ (Mod p)** You take the exponent of the number **a** and calculate the result **(Mod p)**:

Examples of Modular Power Operation:

CASE: p = 7:
$(5)^4$ ≡ 625 ≡ **2 (Mod p)**
CASE: p = 13:
$(5)^4$ ≡ 625 ≡ **1 (Mod p)**

<u>Modular Square Root Operation</u>

A number **n** has two **square roots (Mod p)**, **if and only if** the following condition holds:

n^((p-1)/2) == 1 (Mod p)	**n has two square roots (Mod p)**
n^((p-1)/2) != 1 (Mod p)	**n has no square root Mod p**

If **n** has two **"square roots" (Mod p)**, then there exists two integers **r_1, r_2** such that:
n ≡ r_1^2 (Mod p) and **n ≡ r_2^2 (Mod p)** The main trick is finding the integers **r_1, r_2**. While we **won't** show you how to calculate a **"square root" (Mod p)**, (that's your job!) we will show you how these square roots work:

Examples of Modular Square Root Operation:

CASE: p = 7:
Given below are the two **"square roots" of 4 (Mod p)**
r_1 = Sqrt(**4**) ≡ **2 (Mod p)**
r_2 = Sqrt(**4**) ≡ **5 (Mod p)**
Make sure these are indeed the **"square roots" of 4 (Mod p)**
Check **r_1**: (r_1)^2 ≡ **2 * 2 (Mod p)** ≡ 4 **(Mod p)** ≡ **4 (Mod p)**
Check **r_2**: (r_2)^2 ≡ **5 * 5 (Mod p)** ≡ 25 **(Mod p)** ≡ **4 (Mod p)**
CASE: p = 337639277:
Given below are the two **"square roots" of 17 (Mod p)**
r_1 ≡ Sqrt(**17**) ≡ **113622037 (Mod p)**
r_2 ≡ Sqrt(**17**) ≡ **224017240 (Mod p)**
Make sure these are indeed the **"square roots" of 4 (Mod p)**
Check **r_1**: (r_1)^2 ≡ 12909967292029369 ≡ **17 (Mod p)**
Check **r_2**: (r_2)^2 ≡ 50183723817217600 ≡ **17 (Mod p)**

<u>Modular Division Operation</u>

In order to do modular division, you need to understand the modular multiplicative inverse operation. Assume that **z** is the multiplicative inverse of a number **b** then the following should hold:

$z\ b \equiv 1 \pmod p$

this implies that

$z \equiv (b)^{-1} \pmod p$

thus, $(b)^{-1}$ is the multiplicative inverse of **b**. To divide any number **a** by **b (Mod p)** simply multiply **a** by the multiplicative inverse of **b**

Examples of Modular Division:

CASE: p = 7:

Calculate $5/4 \equiv 5*(4)^{-1} \pmod p$

First, find the inverse of $4 \pmod p$: $(4)^{-1} \equiv 2 \pmod p$

Second, calculate $5*(4)^{-1} \equiv 5 * 2 \equiv 10 \equiv 3 \pmod p$

Check: $(5/4) * 4 \equiv 3 * 4 \equiv 12 \equiv 5 \pmod p$

CASE: p = 13:

Calculate $5/4 \equiv 5*(4)^{-1} \pmod p$

First, find the inverse of $4 \pmod p$: $(4)^{-1} \equiv 10 \pmod p$

Second, calculate $5*(4)^{-1} \equiv 5 * 10 \equiv 50 \equiv 11 \pmod p$

Check: $(5/4) * 4 \equiv 11 * 4 \equiv 44 \equiv 5 \pmod p$

The Program

!!!NOTE!!!: Calculations **may require integers** up to a maximum of **64 bits** in length.

Your task is to write a program that reads quadratic equations from a text file (**a.dat**), and determines whether or not each of the equations in the input has roots **(Mod p)**. Each quadratic equation is on a separate line. The coefficients **a**, **b**, **c** of each quadratic equation and a modulus **p** are given on each line. You may safely assume that all the non−negative values of **p** are odd prime numbers, however, if you encounter a negative **p** value, you should output the message **"invalid input"** as shown below in the sample. Your **program must be efficient**, because the **input file will contain a large number of equations** to solve. For each equation, output the equation and the root(s) in the following format:

Q[x_] := Mod[***a***x^2 + ***b***x + ***c***, ***p***]
{ ***root(s) or message goes here*** }
...blank line...
Q[x_] := Mod[***a***x^2 + ***b***x + ***c***, ***p***]
{ ***root(s) or message goes here*** }
...blank line...

To see how this format corresponds to actual input look at the sample input and output given below.

Sample Input	Sample Output
4 3 3 -13 4 3 3 13 17 8 1 71 3 1000 65709 337639 1 179344794 146367396 179424691	Q[x_] := Mod[4x^2 + 3x + 3, -13] { invalid input } Q[x_] := Mod[4x^2 + 3x + 3, 13] { 11 } Q[x_] := Mod[17x^2 + 8x + 1, 71] { has no roots } Q[x_] := Mod[3x^2 + 1000x + 65709, 337639] { 2345, 109868 } Q[x_] := Mod[1x^2 + 179344794x + 146367396, 179424691] { 78021, 1876 }