Modular Roots

You probably know how to find the roots of a *complex* number. For example, the fourth roots of -4 are $1+i$, $1-i$, $-1+i$, and $-1-i$. In fact, one can prove (using the *Fundamental Theorem of Algebra*) that every nonzero complex number has *exactly* k complex k^{th} roots.

We can also find roots of a number modulo p. A number, x, is a k^{th} root of n modulo p if $x^k \equiv n \pmod{p}$. For example, the fourth roots of the number -4 modulo 29 are 11, 13, 16, and 18; you can check that $11^4 \equiv 13^4 \equiv 16^4 \equiv 18^4 \equiv -4 \pmod{29}$, and no other distinct values satisfy this. Also, the fourth roots of 4 modulo 23 are 5 and 18. Clearly, the number of k^{th} roots modulo p is no longer k.

Observe that there's a slight technicality here: if x is a k^{th} root of n modulo p, then so is $x \pm bp$ for any integer b. For example, the cube roots of 2 modulo 11 are 7, 18, 29, 40, $\dots$, and also -4, -15, -26, -37, $\dots$. We want to count all of these only once because they are all congruent modulo 11 so, for our purposes, we only count two roots as distinct if they are *not* congruent modulo p.

Task

Given a prime p and integers k and n, find *all* the k^{th} roots of n modulo p. To avoid double counting, only consider values in the set $\{0, 1, 2, \ldots, p-1\}$.

To make the problem a little more challenging, there will be multiple test cases within a test file, but the prime p is fixed for each test case within a test file.

Resources

- Fermat's Little Theorem
- Modular Arithmetic and Exponents
- Primitive Roots

Input Format

The first line contains two space-separated integers, p and Q, respectively.
Each of the Q subsequent lines describes a test case in the form of two space-separated integers, k and n, respectively.

Constraints

- $2 \le p \le 5\times 10^6$
- $1 \le Q \le 10^5$
- $1 \le k \le 10^5$
- $|n| \le 5\times 10^6$
- *The number of output values per test file is* $\le 6 \times 10^5.$

Output Format

For each test case, print a single line containing all the k^{th} roots of n modulo p as a series of space-separated integers in increasing order; if there are no k^{th} roots, print `NONE`.

Sample Input

```
29 3
4 -4
7 10
111 32
```

Sample Output

```
11 13 16 18
NONE
10
```