10870 Recurrences

Consider recurrent functions of the following form:

$$f(n) = a_1 f(n-1) + a_2 f(n-2) + a_3 f(n-3) + \ldots + a_d f(n-d), \text{ for } n > d,$$

where a_1, a_2, ..., a_d are arbitrary constants.

A famous example is the Fibonacci sequence, defined as: $f(1) = 1$, $f(2) = 1$, $f(n) = f(n-1) + f(n-2)$. Here $d = 2$, $a_1 = 1$, $a_2 = 1$.

Every such function is completely described by specifying d (which is called the order of recurrence), values of d coefficients: a_1, a_2, ..., a_d, and values of $f(1)$, $f(2)$, ..., $f(d)$. You'll be given these numbers, and two integers n and m. Your program's job is to compute $f(n)$ modulo m.

Input

Input file contains several test cases. Each test case begins with three integers: d, n, m, followed by two sets of d non-negative integers. The first set contains coefficients: a_1, a_2, ..., a_d. The second set gives values of $f(1)$, $f(2)$, ..., $f(d)$.

You can assume that: $1 \le d \le 15$, $1 \le n \le 2^{31} - 1$, $1 \le m \le 46340$. All numbers in the input will fit in signed 32-bit integer.

Input is terminated by line containing three zeroes instead of d, n, m. Two consecutive test cases are separated by a blank line.

Output

For each test case, print the value of $f(n) (\bmod m)$ on a separate line. It must be a non-negative integer, less than m.

Sample Input

```
1 1 100
2
1

2 10 100
1 1
1 1

3 2147483647 12345
12345678 0 12345
1 2 3

0 0 0
```

Sample Output

```
1
55
423
```