F. Test Data Generation

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Test data generation is not an easy task! Often, generating big random test cases is not enough to ensure thorough testing of solutions for correctness.

For example, consider a problem from an old Codeforces round. Its input format looks roughly as follows:

*The first line contains a single integer n ($1 \le n \le max_n$) — the size of the set. The second line contains n distinct integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le max_a$) — the elements of the set **in increasing order**.*

If you don't pay attention to the problem solution, it looks fairly easy to generate a good test case for this problem. Let $n = max_n$, take random distinct a_i from 1 to max_a, sort them... Soon you understand that it's not that easy.

Here is the actual problem solution. Let g be the greatest common divisor of $a_1, a_2, ..., a_n$. Let $x = a_n / g - n$. Then the correct solution outputs "`Alice`" if x is odd, and "`Bob`" if x is even.

Consider two wrong solutions to this problem which differ from the correct one only in the formula for calculating x.

The first wrong solution calculates x as $x = a_n / g$ (without subtracting n).

The second wrong solution calculates x as $x = a_n - n$ (without dividing by g).

A test case is interesting if it makes **both** wrong solutions output an incorrect answer.

Given max_n, max_a and q, find the number of interesting test cases satisfying the constraints, and output it modulo q.

Input

The only line contains three integers max_n, max_a and q ($1 \le max_n \le 30\,000$; $max_n \le max_a \le 10^9$; $10^4 \le q \le 10^5 + 129$).

Output

Output a single integer — the number of test cases which satisfy the constraints and make both wrong solutions output an incorrect answer, modulo q.

Examples

input
3 6 100000

output
4

input
6 21 100129

output
154

input
58 787788 50216

output
46009

Note

In the first example, interesting test cases look as follows:

```
1
2
```

```
1
4
```

```
1
6
```

```
3
2 4 6
```