Problem D
Magic Power

Time limit: 3 seconds
Memory limit: 1024 megabytes

Problem Description

Morris is experimenting with an ancient magical spell that requires determine the power of a number, but not just any power.

The magic formula is:
$result = (base^{exp})\%MOD$

However, the number exp can be extremely large, so large that trying to directly compute $base^{exp}$ before taking the modulo will cause the number to grow beyond imagination (and your program will run out of time and memory).

Your task is to determine this result efficiently and without overflow.

Hint:

- Avoid overflow:
  Computing $base^{exp}$ directly is impossible for large exp because the number grows too fast. Instead, use the property:

  $(a \times b)\%m = [(a\%m) \times (b\%m)]\%m$

  This allows you to apply $\%\ MOD$ at each multiplication step without affecting the final result.

- Optimize the time complexity:
  A naive loop that multiplies $base$ exp times is $\mathcal{O}(exp)$, which is far too slow when exp is large. Use Binary Exponentiation (also known as Fast Power) to reduce the time complexity to $\mathcal{O}(log exp)$ by using the following rules:

  - If exp is even: $base^{exp} = (base^{exp/2})^2$
  - If exp is odd: $base^{exp} = base^1 \times (base^{exp/2})^2$

  Apply $\%\ MOD$ after each multiplication to keep numbers small.

- Recursive thinking:
  This method naturally fits a divide-and-conquer recursive approach: at each step, you solve the smaller problem $base^{exp/2}$ first, then combine the results according to whether exp is even or odd.

Input Format

The first line contains an integer T, representing the number of test cases $(1 \leq T \leq 10,000)$. The next T lines each contain three integers, $base(1 \leq base \leq 10^9)$, $exp(0 \leq exp \leq 10^{18})$ and $MOD(1 \leq MOD \leq 10^9)$, separated by a space.

Output Format

For each test case, output the $result$.

Technical Specification

- $1 \leq T \leq 10,000$
- $1 \leq base \leq 10^9$
- $0 \leq exp \leq 10^{18}$
- $1 \leq MOD \leq 10^9$

Sample Input 1

```
3
2 10 9999
2 10 1000
3 50 1000000000
```

Sample Output 1

```
1024
24
588770249
```