# Modular Tetration

The ordinary arithmetical operations of addition, multiplication and exponentiation are naturally extended into a sequence of hyperoperations as follows.

3×7  = 3 + 3 + 3 + 3 + 3 + 3 + 3 ; there are 7 copies of "3"
3^7  = 3 × 3 × 3 × 3 × 3 × 3 × 3 ; there are 7 copies of "3"
3^^7 = (3^(3^(3^(3^(3^(3^3)))))) ; there are 7 copies of "3"

To extend the sequence of operations beyond exponentiation, Knuth defined a “double arrow” operator to denote iterated exponentiation (tetration) ^^ in ASCII notation.
This gives us some very big numbers, your task will be to compute modular tetration.
***X*^0=1** for all ***X***, as an empty product. ***X*^^0=1** for all ***X***, for similar reason.

## Input

The first line of input contains an integer ***T***, the number of test cases.
On each of the next ***T*** lines, your are given three integers ***X***, ***N***, and ***M***.

## Output

For each test case, you have to print ***X^^N modulo M***.

## Example

**Input:**
3
3 2 20
3 3 345
993306745 75707320 1000000000

**Output:**
7
312
884765625

## Constraints

0 < T <= 10^4
X, N, M unsigned 32bit integers
1 < M

## Explanations

3^^2 = 3^3 = 27 = 7 [mod 20]
3^^3 = 3^(3^3) = 3^27 = 7625597484987 = 312 [mod 345]

Problem designed to be solvable using some 'slow' languages like Python, under half the time limit.
It is recommended to solve first Power Tower City.
;-) Have fun.