Balance of Brahan Again

This problem is back again!

Kevin put all his money into Brahan Bank. Why? This bank is so awesome that for each day, the balance is **multiplied** by r! For example, if you put 100 Birr in this bank, the next day the balance becomes $100r$, and two days after the balance becomes $100r^2$, and so on.

Kevin is so happy. He put b Birr into the bank, and he is planning to withdraw all the money after n days. However, to withdraw money, it is necessary to press the amount of money in the ATM. So Kevin wants to know the last 9 digits (in decimal system) of the balance after n days.

Given b, r and n, write a program that calculates the last 9 digits of the balance after n days.

Input

Your input consists of an arbitrary number of lines, but no more than 100.

Each line contains three integers b $(1 \le b < 10^9)$, r $(1 \le r < 10^9)$ and n $(1 \le n \le 10^{18})$, which means you have to calculate the last 9 digits of the balance. Please use 64-bit integers to read and store n.

The end of input is indicated by a line containing only the value -1.

Output

For each line, print the answer. You should print **exactly** 9 digits. If the balance is less than 9 digits, print '0' like the example below.

Example

Standard input	Standard output
50 2 1 2 2 2 19 49 121 1 3 1000000000000000000 9 6 54 -1	000000100 000000008 545264931 000000001 325937664

Time Limit

1 second.

Hints

How to calculate the last two digits of 913913×141299?

1. $913913 \times 141299 = 129134992987$. So we know that the last two digits are '87'.
2. We just need to consider two digits of 913913 and 141299. So we can calculate $13 \times 99 = 1287$ instead, and know that the last two digits are '87'.

Like this, while calculating the last 9 digits of r^n by multiplying some numbers, we can just store the last 9 digits during the calculation. Try to use the modulo operator ('%') to do that.

Also, $99999999 \times 99999999 = 9999999980000000001 > 2^{31}$, so you might need to use 64-bit integer types during the calculation.

For your convenience, we provide the code that calculates the answer in $O(n)$:

```
int balance (int b, int r, int n) {
  if(n == 0) {
    return b % 1000000000;
  }else {
    return ((long long)balance(b, r, n-1) * r) % 1000000000;
  }
}
```