Insane Exponentiation 1

This problem was worth 500 points and was ranked easy. The authors of this problem are Aryan V S and Dhruv Chawla.

Note: GitHub does not support LaTex in Markdown. If you want a more readable version of the problem, download the PDF file instead.

Story

The problem is named "Insane Exponentiation 1" because we initially had another challenge related to exponentiation called "Insane Exponentiation 2" in the Hard category. It turned out to be very hard to solve ourselves and people we asked for help, and we decided to not spend more time on it and come up with a different problem.

Statement

Aryan likes number theory. So, Dhruv decided to come up with a challenge for him to solve.

Dhruv gives Aryan three integers b, e and p. He wants Aryan to find the last p digits of the value b^e. If the value has lesser than p digits, pad the value with leading zeroes.

Since Aryan does not know how to approach the problem, he asks for your help. You are required to answer t testcases.

Input Format

The first line of the input contains an integer t - the number of testcases.

The following t lines each contain three space separated integers b, e and p.

Output Format

The output should contain t lines.

The i^{th} line should contain the answer to the i^{th} testcase.

Constraints

$1 \leq t \leq 100000$

$1 \leq b \leq 10^9$

$1 \leq e \leq 10^9$

$1 \leq p \leq 9$

Sample Tests

Sample Test 1

Input

```
5
5 9 2
4 6 5
4 1 1
7 5 2
10 6 5
```

Output

```
25
04096
4
07
00000
```

Explanation

In the first testcase, $5^9 = 1953125$. The last 2 digits are 25.

In the second testcase, $4^6 = 4096$. The last 5 digits are 04096 (notice the leading padding because the number of digits in the result is less than 5).

In the third testcase, $4^1 = 4$. The last 1 digits are 4.

In the fourth testcase, $7^5 = 16807$. The last 2 digits are 07.

In the fifth testcase, $10^6 = 1000000$. The last 5 digits are 00000.

Solution

This problem requires knowing a simple trick from math. When we only care about the last few digits of the product/sum/(any other similar arithmetic operation) of numbers, we can discard the remaining prefix of digits and perform operations only on the ones we care about. For example, if we wanted to know the last digit of the product of $9817436 * 42424242$, we do not need to care about the anything but the last digit in the product, which is simply $6 * 2 = 12 \rightarrow 2$.

For the last p digits, we only need to perform the products for p digits at a time and can get rid of the remaining prefix of digits. This can be easily done using modulo operation with mod as 10^p.

We also require a fast method to calculate powers of a number. The technique we can use for the same is called Binary Exponentiation.

Solution in C++:

```cpp
#include <bits/stdc++.h>

constexpr bool test_cases = true;
```

```cpp

int64_t binary_exponentiate (int64_t a, int64_t b, int64_t mod) {
  a %= mod;
  int64_t r = 1;

  while (b > 0) {
    if (b & 1)
      r = r * a % mod;
    a *= a;
    a %= mod;
    b /= 2;
  }

  return r;
}

void solve () {
  int64_t b, e, p;
  std::cin >> b >> e >> p;

  int64_t p10 = 1;
  for (int i = 0; i < p; ++i)
    p10 *= 10;

  int64_t r = binary_exponentiate(b, e, p10);
  std::string s = std::to_string(r);

  int padding = p - s.length();

  if (padding > 0)
    s = std::string(padding, '0') + s;

  std::cout << s << '\n';
}

int main () {
  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int cases = 1;
  if (test_cases)
    std::cin >> cases;
  while (cases--)
    solve();

  return 0;
}
```

Solution in Python:

```python
t = int(input())

for _ in range(t):
  b, e, p = map(int, input().split())
  y = str(pow(b, e, 10 ** p))

  padding = p - len(y)
  y = '0' * padding + y

  print(y)
```