The fibonacci number is defined by the following recurrence:

- $fib(0) = 0$
- $fib(1) = 1$
- $fib(n) = fib(n-1) + fib(n-2)$

But we're not interested in the fibonacci numbers here. We would like to know how many calls does it take to evaluate the n-th fibonacci number if we follow the given recurrence. Since the numbers are going to be quite large, we'd like to make the job a bit easy for you. We'd only need the last digit of the number of calls, when this number is represented in base b.

Input

Input consists of several test cases. For each test you'd be given two integers n ($0 \leq n < 2^{63} - 1$), b ($0 < b \leq 10000$). Input is terminated by a test case where $n = 0$ and $b = 0$, you must not process this test case.

Output

For each test case, print the test case number first. Then print n, b and the last digit (in base b) of the number of calls. There would be a single space in between the two numbers of a line.

Note that the last digit has to be represented in decimal number system.

Sample Input

```
0 100
1 100
2 100
3 100
10 10
0 0
```

Sample Output

```
Case 1: 0 100 1
Case 2: 1 100 1
Case 3: 2 100 3
Case 4: 3 100 5
Case 5: 10 10 7
```