Problem I. Bubble Sort

Time limit: 10 seconds

Bubble sort is a simple sorting algorithm that repeatedly steps through the list to be sorted, compares each pair of adjacent items and swaps them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.

Izdihar is an ACM-ICPC master and provides a pseudocode implementation about the bubble sort for you. The algorithm for a list of sortable items A can be expressed as (1-based array):

```
1: function BubbleSort(A)
2:     n ← the length of A
3:     k ← n − 1                    ▷ note the initial value of k might be replaced by a given number
4:     for step = 1 to k do         ▷ passing through the list k times
5:         for i = 2 to n do
6:             if A[i − 1] > A[i] then
7:                 swap A[i − 1] and A[i]
```

She says a permutation of 1 to n is almost sorted if the length of its longest increasing subsequence is at least $n-1$.

You are asked to count the number of permutations of 1 to n which, after k times passing through the list in the bubble sort, would become an almost sorted permutation.

Input

The input contains several test cases, and the first line is a positive T indicating the number of test cases which is up to 5000.

For each test case, a line contains three integers n, k ($1 \leq n, k \leq 50$) which are described as above, and q ($10^8 \leq q \leq 10^9$) which is a prime number for the output.

Output

For each test case, output a line containing `Case #x: y`, where `x` is the test case number starting from 1. And `y` is the remainder of the number of permutations which meet the requirement, divided by q.

Sample

standard input	standard output
5 5 1 998244353 5 2 998244353 5 3 998244353 5 4 998244353 5 5 998244353	Case #1: 74 Case #2: 114 Case #3: 120 Case #4: 120 Case #5: 120