Round B APAC Test 2017

A. Sherlock and Parentheses

B. Sherlock and Watson Gym Secrets

C. Watson and Intervals

D. Sherlock and Permutation Sorting

Questions asked 1

Submissions

Sherlock and Parentheses

4pt Not attempted **3846/5689 users** correct (68%)

7pt Not attempted **2912/3801 users** correct (77%)

Sherlock and Watson Gym Secrets

6pt Not attempted **1760/3710 users** correct (47%)

15pt Not attempted **268/1026 users** correct (26%)

Watson and Intervals

8pt Not attempted **526/1376 users** correct (38%)

17pt Not attempted **152/284 users** correct (54%)

Sherlock and Permutation Sorting

19pt Not attempted **44/428 users** correct (10%)

24pt Not attempted **15/27 users** correct (56%)

Top Scores

bsbandme	100
alecsyde	100
RiverBlessPeople	100
NAFIS	100
izrak	100
dragon7	100
winoros	100
gvaibhav21	100
stonebuddha	100
VastoLorde95	100

Problem D. **Sherlock and Permutation Sorting**

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input 19 points	Solve D-small
Large input 24 points	Solve D-large

Problem

Sherlock and Watson have already been introduced to sorting in their computer programming course. Now, Watson has always been curious about parallel computing and wants to sort a permutation of the integers 1 through **N** by breaking it into chunks, sorting the chunks individually, and then concatenating them.

For a permutation $p_1, p_2, \ldots, p_N$, a chunk is a contiguous subarray of the permutation: i.e., a sequence of elements $p_i, p_{i+1}, \ldots, p_j$, for the elements at indexes i and j such that $1 \le i \le j \le \mathbf{N}$.

Watson wants to partition his permutation into an ordered list of one or more chunks, without changing the order that the elements are in, in such a way that each element of the permutation is in exactly one chunk, and all elements in a chunk are smaller than all elements in any later chunk.
For example, for the permutation [2, 1, 3, 5, 4], these are the only four legal ways for Watson to break it into chunks: [[2, 1, 3], [5, 4]] or [[2, 1], [3, 5, 4]] or [[2, 1], [3], [5, 4]] or [[2, 1, 3, 5, 4]]. Watson is happiest when there are as many chunks as possible; we denote the maximum number of chunks for a permutation p as f(p). In this example, the maximum number of chunks is 3.

Watson wants to consider all permutations p of the numbers 1 through **N**, and find the **sum of squares** of f(p). Watson knows Sherlock might come in handy and comes to him for help, but Sherlock is as clueless as Watson and asks you for help. As the sum of squares can be large, please find it modulo **M**.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case consists of one line with two integers **N** and **M**.

Output

For each test case, output one line containing `Case #x: y`, where x is the test case number (starting from 1) and y is the sum of squares of `f(p)` for all permutations p of size **N**, modulo **M**.

Limits

$1 \le \mathbf{M} \le 10^9$.

Small dataset

$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{N} \le 100$.

Large dataset

$1 \le \mathbf{T} \le 20$.
$1 \le \mathbf{N} \le 5000$.

Sample

Input	Output
3 1 2 2 4 3 7	Case #1: 1 Case #2: 1 Case #3: 6

In Case 1, there is only one permutation. `f([1]) * f([1]) % 2` = 1.

In Case 2, there are two permutations.
`f([1, 2])` = 2.
`f([2, 1])` = 1.
$(2^2 + 1^2)$ % 4 = 1.

In Case 3, there are six permutations.
`f([1, 2, 3])` = 3.
`f([1, 3, 2])` = 2.
`f([2, 1, 3])` = 2.
`f([2, 3, 1])` = 1.
`f([3, 1, 2])` = 1.
`f([3, 2, 1])` = 1.
$(3^2 + 2^2 + 2^2 + 1^2 + 1^2 + 1^2) \% 7$ = 6.