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 A. Sherlock and Parentheses

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 4 points	Solve A-small
Large input 7 points	Solve A-large

Problem

Sherlock and Watson have recently enrolled in a computer programming course. Today, the tutor taught them about the balanced parentheses problem. A string S consisting only of characters `(` and/or `)` is *balanced* if:

- It is the empty string, or:
- It has the form `(`S`)`, where S is a balanced string, or:
- It has the form S_1S_2, where S_1 is a balanced string and S_2 is a balanced string.

Sherlock coded up the solution very quickly and started bragging about how good he is, so Watson gave him a problem to test his knowledge. He asked Sherlock to generate a string S of **L** + **R** characters, in which there are a total of **L** left parentheses `(` and a total of **R** right parentheses `)`. Moreover, the string must have as many different balanced non-empty substrings as possible. (Two substrings are considered different as long as they start or end at different indexes of the string, even if their content happens to be the same). Note that S itself does not have to be balanced.

Sherlock is sure that once he knows the maximum possible number of balanced non-empty substrings, he will be able to solve the problem. Can you help him find that maximum number?

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: **L** and **R**.

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 answer, as described above.

Limits

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

Small dataset

$0 \le \mathbf{L} \le 20$.
$0 \le \mathbf{R} \le 20$.
$1 \le \mathbf{L} + \mathbf{R} \le 20$.

Large dataset

$0 \le \mathbf{L} \le 10^5$.
$0 \le \mathbf{R} \le 10^5$.
$1 \le \mathbf{L} + \mathbf{R} \le 10^5$.

Sample

Input	Output
3 1 0 1 1 3 2	Case #1: 0 Case #2: 1 Case #3: 3

In Case 1, the only possible string is `(`. There are no balanced non-empty substrings.
In Case 2, the optimal string is `()`. There is only one balanced non-empty substring: the entire string itself.
In Case 3, both strings `()()(` and `(()()` give the same optimal answer.
For the case `()()(`, for example, the three balanced substrings are `()` from indexes 1 to 2, `()` from indexes 3 to 4, and `()()` from indexes 1 to 4.

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 B. Sherlock and Watson Gym Secrets

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 6 points	Solve B-small
Large input 15 points	Solve B-large

Problem

Watson and Sherlock are gym buddies.

Their gym trainer has given them three numbers, **A**, **B**, and **N**, and has asked Watson and Sherlock to pick two different **positive integers** i and j, where i and j are both less than or equal to **N**. Watson is expected to eat exactly $i^{\mathbf{A}}$ sprouts every day, and Sherlock is expected to eat exactly $j^{\mathbf{B}}$ sprouts every day.

Watson and Sherlock have noticed that if the total number of sprouts eaten by them on a given day is divisible by a certain integer **K**, then they get along well that day.

So, Watson and Sherlock need your help to determine how many such pairs of **(i, j)** exist, where i != j. As the number of pairs can be really high, please output it modulo $\mathbf{10^9+7}$ **(1000000007)**.

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 4 integers **A**, **B**, **N** and **K**, as described above.

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 required answer.

Limits

$1 \le \mathbf{T} \le 100$.
$0 \le \mathbf{A} \le 10^6$.
$0 \le \mathbf{B} \le 10^6$.

Small dataset

$1 \le \mathbf{K} \le 10000$.
$1 \le \mathbf{N} \le 1000$.

Large dataset

$1 \le \mathbf{K} \le 100000$.
$1 \le \mathbf{N} \le 10^{18}$.

Sample

Input	Output
3 1 1 5 3 1 2 4 5 1 1 2 2	Case #1: 8 Case #2: 3 Case #3: 0

In Case 1, the possible pairs are (1, 2), (1, 5), (2, 1), (2, 4), (4, 2), (4, 5), (5, 1), and (5, 4).
In Case 2, the possible pairs are (1, 2), (1, 3), and (4, 1).
In Case 3, No possible pairs are there, as i != j.

Powered by

Google Cloud Platform

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 C. Watson and Intervals

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 8 points	Solve C-small
Large input 17 points	Solve C-large

Problem

Sherlock and Watson have mastered the intricacies of the language C++ in their programming course, so they have moved on to algorithmic problems. In today's class, the tutor introduced the problem of merging one-dimensional intervals. **N** intervals are given, and the `i`th interval is defined by the inclusive endpoints [$\mathbf{L_i}$, $\mathbf{R_i}$], where $\mathbf{L_i} \le \mathbf{R_i}$.

The tutor defined the *covered area* of a set of intervals as the number of integers appearing in at least one of the intervals. (Formally, an integer p contributes to the covered area if there is some j such that $\mathbf{L_j} \le p \le \mathbf{R_j}$.)

Now, Watson always likes to challenge Sherlock. He has asked Sherlock to remove exactly one interval such that the covered area of the remaining intervals is minimized. Help Sherlock find this minimum possible covered area, after removing exactly one of the **N** intervals.

Input

Each test case consists of one line with eight integers **N**, $\mathbf{L_1}$, $\mathbf{R_1}$, **A**, **B**, $\mathbf{C_1}$, $\mathbf{C_2}$, and **M**. **N** is the number of intervals, and the other seven values are parameters that you should use to generate the other intervals, as follows:

First define $x_1 = \mathbf{L_1}$ and $y_1 = \mathbf{R_1}$. Then, use the recurrences below to generate x_i, y_i for `i` = 2 to **N**:

- x_i = ($\mathbf{A}*x_{i-1} + \mathbf{B}*y_{i-1} + \mathbf{C_1}$) modulo **M**.
- y_i = ($\mathbf{A}*y_{i-1} + \mathbf{B}*x_{i-1} + \mathbf{C_2}$) modulo **M**.

We define $\mathbf{L_i}$ = `min(`x_i`,` y_i`)` and $\mathbf{R_i}$ = `max(`x_i`,` y_i`)`, for all `i` = 2 to **N**.

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 minimum possible covered area of all of the intervals remaining after removing exactly one interval.

Limits

$1 \le \mathbf{T} \le 50$.
$0 \le \mathbf{L_1} \le \mathbf{R_1} \le 10^9$.
$0 \le \mathbf{A} \le 10^9$.
$0 \le \mathbf{B} \le 10^9$.
$0 \le \mathbf{C_1} \le 10^9$.
$0 \le \mathbf{C_2} \le 10^9$.
$1 \le \mathbf{M} \le 10^9$.

Small dataset

$1 \le \mathbf{N} \le 1000$.

Large dataset

$1 \le \mathbf{N} \le 5 * 10^5 (500000)$.

Sample

Input	Output
3 1 1 1 1 1 1 1 1 3 2 5 1 2 3 4 10 4 3 4 3 3 8 10 10	Case #1: 0 Case #2: 4 Case #3: 9

In case 1, using the generation method, the set of intervals generated are: {[1, 1]}. Removing the only interval, the *covered area* is 0.

In case 2, using the generation method, the set of intervals generated are: {[2, 5], [3, 5], [4, 7]}. Removing the first, second or third interval would cause the covered area of remaining intervals to be 5, 6 and 4, respectively.

In case 3, using the generation method, the set of intervals generated are: {[3, 4], [1, 9], [0, 8], [2, 4]}. Removing the first, second, third or fourth interval would cause the covered area of remaining intervals to be 10, 9, 9 and 10, respectively.

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 $1 \le i \le j \le N$.

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.
$(3^2 + 2^2 + 2^2 + 1^2 + 1^2 + 1^2) \% 7 = 6$.
