A. Cards Partition

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

DJ Genki vs Gram - Einherjar Joker

You have some cards. An integer between 1 and n is written on each card: specifically, for each i from 1 to n, you have a_i cards which have the number i written on them.

There is also a shop which contains unlimited cards of each type. You have k coins, so you can buy **at most** k new cards in total, and the cards you buy can contain any integer **between 1 and n**, inclusive.

After buying the new cards, you must partition **all** your cards into decks, according to the following rules:

- all the decks must have the same size;
- there are no pairs of cards with the same value in the same deck.

Find the maximum possible size of a deck after buying cards and partitioning them optimally.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n, k ($1 \le n \le 2 \cdot 10^5, 0 \le k \le 10^{16}$) — the number of distinct types of cards and the number of coins.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^{10}, \sum a_i \ge 1$) — the number of cards of type i you have at the beginning, for each $1 \le i \le n$.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum possible size of a deck if you operate optimally.

Standard Input	Standard Output
9 3 1 3 2 2 5 4 2 6 1 2 4 2 100 1410065408 10000000000 10 8 7 4 6 6 9 3 10 2 8 7 2 12 2 2 2 70 0 1 1 0	2 3 1 7 2 2 1 1 2

1 3 0 2 1 2 3 1 0 3 3	

Note

In the first test case, you can buy one card with the number 1, and your cards become $[1, 1, 1, 1, 2, 2, 3, 3]$. You can partition them into the decks $[1, 2]$, $[1, 2]$, $[1, 3]$, $[1, 3]$: they all have size 2, and they all contain distinct values. You can show that you cannot get a partition with decks of size greater than 2, so the answer is 2.

In the second test case, you can buy two cards with the number 1 and one card with the number 3, and your cards become $[1, 1, 1, 1, 2, 2, 2, 2, 2, 3, 3, 4, 4, 5, 5, 5, 5]$, which can be partitioned into $[1, 2, 3]$, $[1, 2, 4]$, $[1, 2, 5]$, $[1, 2, 5]$, $[2, 3, 5]$, $[2, 4, 5]$. You can show that you cannot get a partition with decks of size greater than 3, so the answer is 3.

,

B. Speedbreaker

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Djjaner - Speedbreaker

There are n cities in a row, numbered $1, 2, \ldots, n$ left to right.

- At time 1, you conquer exactly one city, called the *starting city*.
- At time $2, 3, \ldots, n$, you can choose a city adjacent to the ones conquered so far and conquer it.

You win if, for each i, you conquer city i at a time no later than a_i. A winning strategy may or may not exist, also depending on the starting city. How many starting cities allow you to win?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of cities.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the deadlines for conquering the cities.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the number of starting cities that allow you to win.

Standard Input	Standard Output
3 6 6 3 3 3 5 5 6 5 6 4 1 4 5 9 8 6 4 2 1 3 5 7 9	3 0 1

Note

In the first test case, cities 2, 3, and 4 are good starting cities.

In the second test case, there are no good starting cities.

In the third test case, the only good starting city is city 5.

,

C. Tree Pruning

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

t+pazolite, ginkiha, Hommarju - Paved Garden

You are given a tree with n nodes, rooted at node 1. In this problem, a leaf is a non-root node with degree 1.

In one operation, you can remove a leaf and the edge adjacent to it (possibly, new leaves appear). What is the minimum number of operations that you have to perform to get a tree, also rooted at node 1, where all the leaves are at the same distance from the root?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($3 \le n \le 5 \cdot 10^5$) — the number of nodes.

Each of the next $n-1$ lines contains two integers u, v ($1 \le u, v \le n$, $u \neq v$), describing an edge that connects u and v. It is guaranteed that the given edges form a tree.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$.

Output

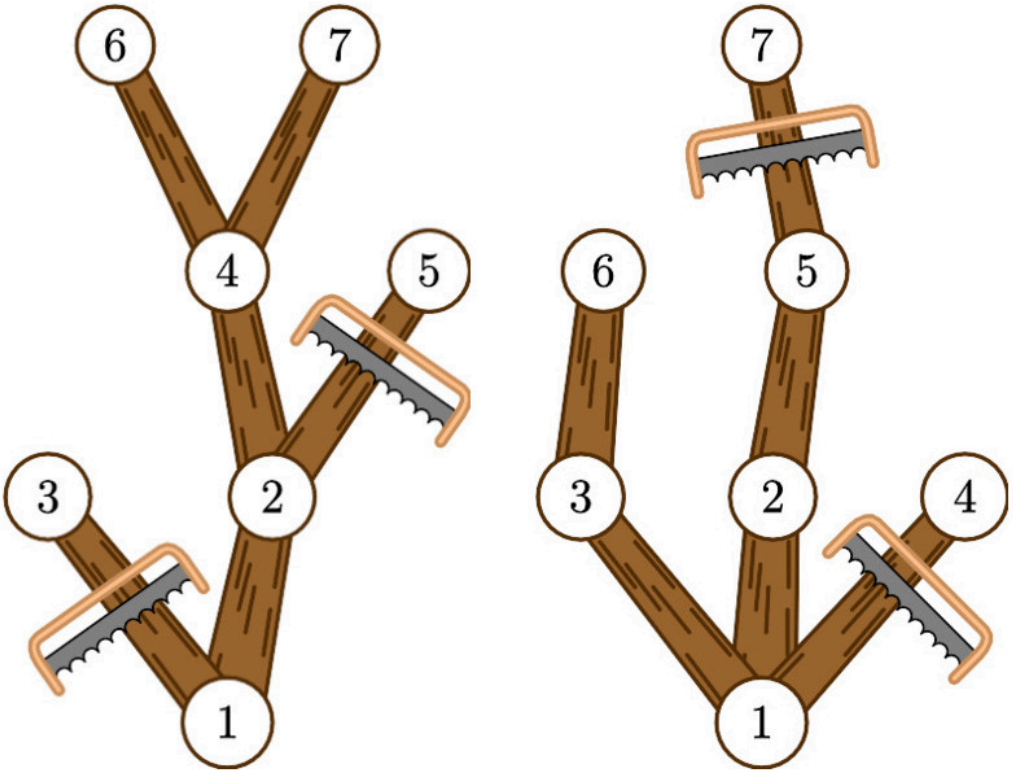
For each test case, output a single integer: the minimum number of operations needed to achieve your goal.

Standard Input	Standard Output
3 7 1 2 1 3 2 4 2 5 4 6 4 7 7 1 2 1 3 1 4 2 5 3 6 5 7 15 12 9 1 6 6 14 9 11	2 2 5

8 7 3 5 13 5 6 10 13 15 13 6 14 12 7 2 8 1 1 4	

Note

In the first two examples, the tree is as follows:



In the first example, by removing edges $(1, 3)$ and $(2, 5)$, the resulting tree has all leaves (nodes 6 and 7) at the same distance from the root (node 1), which is 3. The answer is 2, as it is the minimum number of edges that need to be removed to achieve the goal.

In the second example, removing edges $(1, 4)$ and $(5, 7)$ results in a tree where all leaves (nodes 4 and 5) are at the same distance from the root (node 1), which is 2.

,

D. Max Plus Min Plus Size

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

EnV - The Dusty Dragon Tavern

You are given an array $a_1, a_2, \ldots, a_n$ of positive integers.

You can color some elements of the array red, but there cannot be two adjacent red elements (i.e., for $1 \leq i \leq n-1$, at least one of a_i and a_{i+1} must not be red).

Your score is the maximum value of a red element, plus the minimum value of a red element, plus the number of red elements. Find the maximum score you can get.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the given array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum possible score you can get after coloring some elements red according to the statement.

Standard Input	Standard Output
4 3 5 4 5 3 4 5 4 10 3 3 3 3 4 1 2 3 5 4 10 17 89 92 42 29 41 92 14 70 45	12 11 12 186

Note

In the first test case, you can color the array as follows: $[\color{red}{5}, 4, \color{red}{5}]$. Your score is $\max([5, 5]) + \min([5, 5]) + \text{size}([5, 5]) = 5 + 5 + 2 = 12$. This is the maximum score you can get.

In the second test case, you can color the array as follows: $[4, \color{red}{5}, 4]$. Your score is $\max([5]) + \min([5]) + \text{size}([5]) = 5 + 5 + 1 = 11$. This is the maximum score you can get.

In the third test case, you can color the array as follows: $[\color{red}{3}, 3, \color{red}{3}, 3, \color{red}{4}, 1, 2, \color{red}{3}, 5, \color{red}{4}]$. Your score is $\max([3, 3, 4, 3, 4]) + \min([3, 3, 4, 3, 4]) + \text{size}([3, 3, 4, 3, 4]) = 4 + 3 + 5 = 12$. This is the maximum

score you can get.

,

E1. Complex Segments (Easy Version)

Input file: standard input
Output file: standard output
Time limit: 6 seconds
Memory limit: 256 megabytes

Ken Arai - COMPLEX

This is the easy version of the problem. In this version, the constraints on n and the time limit are lower. You can make hacks only if both versions of the problem are solved.

A set of (closed) segments is **complex** if it can be partitioned into some subsets such that

- all the subsets have the same size; and
- a pair of segments intersects **if and only if** the two segments are in the same subset.

You are given n segments $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$. Find the maximum size of a **complex** subset of these segments.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^4$) — the number of segments.

The second line of each test case contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 2n$) — the left endpoints of the segments.

The third line of each test case contains n integers $r_1, r_2, \ldots, r_n$ ($l_i \le r_i \le 2n$) — the right endpoints of the segments.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^4$.

Output

For each test case, output a single integer: the maximum size of a **complex** subset of the given segments.

Standard Input	Standard Output
3 3 1 2 3 5 4 6 5 1 2 3 6 8 5 4 7 9 10 5 3 1 4 1 5 7 2 6 5 10	3 4 4

Note

In the first test case, all pairs of segments intersect, therefore it is optimal to form a single group containing all of the three segments.

In the second test case, there is no valid partition for all of the five segments. A valid partition with four segments is the following: $\{\{[1, 5], [2, 4]\}, \{[6, 9], [8, 10]\}\}$.

In the third test case, it is optimal to make a single group containing all the segments except the second.

,

E2. Complex Segments (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 13 seconds
Memory limit: 256 megabytes

Ken Arai - COMPLEX

This is the hard version of the problem. In this version, the constraints on n and the time limit are higher. You can make hacks only if both versions of the problem are solved.

A set of (closed) segments is **complex** if it can be partitioned into some subsets such that

- all the subsets have the same size; and
- a pair of segments intersects **if and only if** the two segments are in the same subset.

You are given n segments $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$. Find the maximum size of a **complex** subset of these segments.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 3 \cdot 10^5$) — the number of segments.

The second line of each test case contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 2n$) — the left endpoints of the segments.

The third line of each test case contains n integers $r_1, r_2, \ldots, r_n$ ($l_i \le r_i \le 2n$) — the right endpoints of the segments.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, output a single integer: the maximum size of a **complex** subset of the given segments.

Standard Input	Standard Output
3 3 1 2 3 5 4 6 5 1 2 3 6 8 5 4 7 9 10 5 3 1 4 1 5 7 2 6 5 10	3 4 4

Note

In the first test case, all pairs of segments intersect, therefore it is optimal to form a single group containing all of the three segments.

In the second test case, there is no valid partition for all of the five segments. A valid partition with four segments is the following: $\{\{[1, 5], [2, 4]\}, \{[6, 9], [8, 10]\}\}$.

In the third test case, it is optimal to make a single group containing all the segments except the second.

,

F1. Speedbreaker Counting (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   1024 megabytes
```

NightHawk22 - Isolation

This is the easy version of the problem. In the three versions, the constraints on n and the time limit are different. You can make hacks only if all the versions of the problem are solved.

This is the statement of **Problem D1B**:

- There are n cities in a row, numbered $1, 2, \ldots, n$ left to right.
    - At time 1, you conquer exactly one city, called the *starting city*.
    - At time $2, 3, \ldots, n$, you can choose a city adjacent to the ones conquered so far and conquer it.

  You win if, for each i, you conquer city i at a time no later than a_i. A winning strategy may or may not exist, also depending on the starting city. How many starting cities allow you to win?

For each $0 \le k \le n$, count the number of arrays of positive integers $a_1, a_2, \ldots, a_n$ such that

- $1 \le a_i \le n$ for each $1 \le i \le n$;
- the answer to **Problem D1B** is k.

The answer can be very large, so you have to calculate it modulo a given prime p.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 80$). The description of the test cases follows.

The only line of each test case contains two integers n, p ($1 \le n \le 80$, $10^8 \le p \le 10^9$, p is prime) — the number of cities and the modulo.

It is guaranteed that the sum of n over all test cases does not exceed 80.

Output

For each test case, output $n+1$ integers: the i-th integer should be the number of arrays that satisfy the conditions for $k = i - 1$.

Standard Input	Standard Output
11 1 998244353 2 998244353 3 998244353 4 998244353 5 998244353 6 998244353 7 998244353 8 998244353	0 1 1 2 1 14 7 4 2 183 34 19 16 4 2624 209 112 120 48 12 42605 1546 793 992 468 216 36 785910 13327 6556 9190 4672 2880 864 144 16382863 130922 61939 94992 50100 36960 14256 4608 576

9 998244353 10 102275857 10 999662017	382823936 1441729 657784 1086596 583344 488700 216000 96480 23040 2880 20300780 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400 944100756 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400

Note

In the first test case,

- arrays with 1 good starting city: $[1]$.

In the second test case,

- arrays with 0 good starting cities: $[1, 1]$;
- arrays with 1 good starting city: $[1, 2]$, $[2, 1]$;
- arrays with 2 good starting cities: $[2, 2]$.

In the third test case,

- arrays with 0 good starting cities: $[1, 1, 1]$, $[1, 1, 2]$, $[1, 1, 3]$, $[1, 2, 1]$, $[1, 2, 2]$, $[1, 3, 1]$, $[1, 3, 2]$, $[2, 1, 1]$, $[2, 1, 2]$, $[2, 2, 1]$, $[2, 2, 2]$, $[2, 3, 1]$, $[2, 3, 2]$, $[3, 1, 1]$;
- arrays with 1 good starting city: $[1, 2, 3]$, $[1, 3, 3]$, $[2, 1, 3]$, $[3, 1, 2]$, $[3, 1, 3]$, $[3, 2, 1]$, $[3, 3, 1]$;
- arrays with 2 good starting cities: $[2, 2, 3]$, $[2, 3, 3]$, $[3, 2, 2]$, $[3, 3, 2]$;
- arrays with 3 good starting cities: $[3, 2, 3]$, $[3, 3, 3]$.

,

F2. Speedbreaker Counting (Medium Version)

Input file: standard input
Output file: standard output
Time limit: 10 seconds
Memory limit: 1024 megabytes

NightHawk22 - Isolation

This is the medium version of the problem. In the three versions, the constraints on n and the time limit are different. You can make hacks only if all the versions of the problem are solved.

This is the statement of **Problem D1B**:

- There are n cities in a row, numbered $1, 2, \ldots, n$ left to right.
    - At time 1, you conquer exactly one city, called the *starting city*.
    - At time $2, 3, \ldots, n$, you can choose a city adjacent to the ones conquered so far and conquer it.

  You win if, for each i, you conquer city i at a time no later than a_i. A winning strategy may or may not exist, also depending on the starting city. How many starting cities allow you to win?

For each $0 \le k \le n$, count the number of arrays of positive integers $a_1, a_2, \ldots, a_n$ such that

- $1 \le a_i \le n$ for each $1 \le i \le n$;
- the answer to **Problem D1B** is k.

The answer can be very large, so you have to calculate it modulo a given prime p.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The only line of each test case contains two integers n, p ($1 \le n \le 500$, $10^8 \le p \le 10^9$, p is prime) — the number of cities and the modulo.

It is guaranteed that the sum of n over all test cases does not exceed 500.

Output

For each test case, output $n+1$ integers: the i-th integer should be the number of arrays that satisfy the conditions for $k = i - 1$.

Standard Input	Standard Output
11 1 998244353 2 998244353 3 998244353 4 998244353 5 998244353 6 998244353 7 998244353 8 998244353	0 1 1 2 1 14 7 4 2 183 34 19 16 4 2624 209 112 120 48 12 42605 1546 793 992 468 216 36 785910 13327 6556 9190 4672 2880 864 144 16382863 130922 61939 94992 50100 36960 14256 4608 576

9 998244353 10 102275857 10 999662017	382823936 1441729 657784 1086596 583344 488700 216000 96480 23040 2880 20300780 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400 944100756 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400

Note

In the first test case,

- arrays with 1 good starting city: $[1]$.

In the second test case,

- arrays with 0 good starting cities: $[1, 1]$;
- arrays with 1 good starting city: $[1, 2]$, $[2, 1]$;
- arrays with 2 good starting cities: $[2, 2]$.

In the third test case,

- arrays with 0 good starting cities: $[1, 1, 1]$, $[1, 1, 2]$, $[1, 1, 3]$, $[1, 2, 1]$, $[1, 2, 2]$, $[1, 3, 1]$, $[1, 3, 2]$, $[2, 1, 1]$, $[2, 1, 2]$, $[2, 2, 1]$, $[2, 2, 2]$, $[2, 3, 1]$, $[2, 3, 2]$, $[3, 1, 1]$;
- arrays with 1 good starting city: $[1, 2, 3]$, $[1, 3, 3]$, $[2, 1, 3]$, $[3, 1, 2]$, $[3, 1, 3]$, $[3, 2, 1]$, $[3, 3, 1]$;
- arrays with 2 good starting cities: $[2, 2, 3]$, $[2, 3, 3]$, $[3, 2, 2]$, $[3, 3, 2]$;
- arrays with 3 good starting cities: $[3, 2, 3]$, $[3, 3, 3]$.

,

F3. Speedbreaker Counting (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     10 seconds
Memory limit:   1024 megabytes
```

NightHawk22 - Isolation

This is the hard version of the problem. In the three versions, the constraints on n and the time limit are different. You can make hacks only if all the versions of the problem are solved.

This is the statement of **Problem D1B**:

- There are n cities in a row, numbered $1, 2, \ldots, n$ left to right.
    - At time 1, you conquer exactly one city, called the *starting city*.
    - At time $2, 3, \ldots, n$, you can choose a city adjacent to the ones conquered so far and conquer it.

  You win if, for each i, you conquer city i at a time no later than a_i. A winning strategy may or may not exist, also depending on the starting city. How many starting cities allow you to win?

For each $0 \le k \le n$, count the number of arrays of positive integers $a_1, a_2, \ldots, a_n$ such that

- $1 \le a_i \le n$ for each $1 \le i \le n$;
- the answer to **Problem D1B** is k.

The answer can be very large, so you have to calculate it modulo a given prime p.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 3000$). The description of the test cases follows.

The only line of each test case contains two integers n, p ($1 \le n \le 3000$, $10^8 \le p \le 10^9$, p is prime) — the number of cities and the modulo.

It is guaranteed that the sum of n over all test cases does not exceed 3000.

Output

For each test case, output $n+1$ integers: the i-th integer should be the number of arrays that satisfy the conditions for $k = i - 1$.

Standard Input	Standard Output
11 1 998244353 2 998244353 3 998244353 4 998244353 5 998244353 6 998244353 7 998244353 8 998244353	0 1 1 2 1 14 7 4 2 183 34 19 16 4 2624 209 112 120 48 12 42605 1546 793 992 468 216 36 785910 13327 6556 9190 4672 2880 864 144 16382863 130922 61939 94992 50100 36960 14256 4608 576

9 998244353 10 102275857 10 999662017	382823936 1441729 657784 1086596 583344 488700 216000 96480 23040 2880 20300780 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400 944100756 17572114 7751377 13641280 7376068 6810552 3269700 1785600 576000 144000 14400

Note

In the first test case,

- arrays with 1 good starting city: $[1]$.

In the second test case,

- arrays with 0 good starting cities: $[1, 1]$;
- arrays with 1 good starting city: $[1, 2]$, $[2, 1]$;
- arrays with 2 good starting cities: $[2, 2]$.

In the third test case,

- arrays with 0 good starting cities: $[1, 1, 1]$, $[1, 1, 2]$, $[1, 1, 3]$, $[1, 2, 1]$, $[1, 2, 2]$, $[1, 3, 1]$, $[1, 3, 2]$, $[2, 1, 1]$, $[2, 1, 2]$, $[2, 2, 1]$, $[2, 2, 2]$, $[2, 3, 1]$, $[2, 3, 2]$, $[3, 1, 1]$;
- arrays with 1 good starting city: $[1, 2, 3]$, $[1, 3, 3]$, $[2, 1, 3]$, $[3, 1, 2]$, $[3, 1, 3]$, $[3, 2, 1]$, $[3, 3, 1]$;
- arrays with 2 good starting cities: $[2, 2, 3]$, $[2, 3, 3]$, $[3, 2, 2]$, $[3, 3, 2]$;
- arrays with 3 good starting cities: $[3, 2, 3]$, $[3, 3, 3]$.