Codeforces Round #740 (Div. 1, based on VK Cup 2021 - Final (Engine))

A. Charmed by the Game

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alice and Borys are playing tennis.

A tennis match consists of *games*. In each game, one of the players is *serving* and the other one is *receiving*.

Players serve in turns: after a game where Alice is serving follows a game where Borys is serving, and vice versa.

Each game ends with a victory of one of the players. If a game is won by the serving player, it's said that this player *holds serve*. If a game is won by the receiving player, it's said that this player *breaks serve*.

It is known that Alice won a games and Borys won b games during the match. It is unknown who served first and who won which games.

Find all values of k such that exactly k breaks could happen during the match between Alice and Borys in total.

Input

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

Each of the next t lines describes one test case and contains two integers a and b ($0 \le a, b \le 10^5$; $a + b > 0$) — the number of games won by Alice and Borys, respectively.

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

Output

For each test case print two lines.

In the first line, print a single integer m ($1 \le m \le a + b + 1$) — the number of values of k such that exactly k breaks could happen during the match.

In the second line, print m distinct integers $k_1, k_2, \ldots, k_m$ ($0 \le k_1 < k_2 < \ldots < k_m \le a + b$) — the sought values of k **in increasing order**.

Example

input

```
3
2 1
1 1
0 5
```

output

```
4
0 1 2 3
2
0 2
2
2 3
```

Note

In the first test case, any number of breaks between 0 and 3 could happen during the match:

- Alice holds serve, Borys holds serve, Alice holds serve: 0 breaks;
- Borys holds serve, Alice holds serve, Alice breaks serve: 1 break;
- Borys breaks serve, Alice breaks serve, Alice holds serve: 2 breaks;
- Alice breaks serve, Borys breaks serve, Alice breaks serve: 3 breaks.

In the second test case, the players could either both hold serves (0 breaks) or both break serves (2 breaks).

In the third test case, either 2 or 3 breaks could happen:

- Borys holds serve, Borys breaks serve, Borys holds serve, Borys breaks serve, Borys holds serve: 2 breaks;
- Borys breaks serve, Borys holds serve, Borys breaks serve, Borys holds serve, Borys breaks serve: 3 breaks.

B. Up the Strip

time limit per test: 6 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

Note that the memory limit in this problem is lower than in others.

You have a vertical strip with n cells, numbered consecutively from 1 to n from top to bottom.

You also have a token that is initially placed in cell n. You will move the token up until it arrives at cell 1.

Let the token be in cell $x > 1$ at some moment. One shift of the token can have either of the following kinds:

- Subtraction: you choose an integer y between 1 and $x-1$, inclusive, and move the token from cell x to cell $x-y$.
- Floored division: you choose an integer z between 2 and x, inclusive, and move the token from cell x to cell $\lfloor \frac{x}{z} \rfloor$ (x divided by z rounded down).

Find the number of ways to move the token from cell n to cell 1 using one or more shifts, and print it modulo m. Note that if there are several ways to move the token from one cell to another in one shift, all these ways are considered **distinct** (check example explanation for a better understanding).

Input

The only line contains two integers n and m ($2 \le n \le 4 \cdot 10^6$; $10^8 < m < 10^9$; m is a prime number) — the length of the strip and the modulo.

Output

Print the number of ways to move the token from cell n to cell 1, modulo m.

Examples

input
```
3 998244353
```
output
```
5
```

input
```
5 998244353
```
output
```
25
```

input
```
42 998244353
```
output
```
793019428
```

input
```
787788 100000007
```
output
```
94810539
```

Note

In the first test, there are three ways to move the token from cell 3 to cell 1 in one shift: using subtraction of $y=2$, or using division by $z=2$ or $z=3$.

There are also two ways to move the token from cell 3 to cell 1 via cell 2: first subtract $y=1$, and then either subtract $y=1$ again or divide by $z=2$.

Therefore, there are five ways in total.

C. Bottom-Tier Reversals

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You have a permutation: an array $a = [a_1, a_2, \ldots, a_n]$ of distinct integers from 1 to n. The length of the permutation n is odd.

You need to sort the permutation in increasing order.

In one step, you can choose any prefix of the permutation with an odd length and reverse it. Formally, if $a = [a_1, a_2, \ldots, a_n]$, you can choose any odd integer p between 1 and n, inclusive, and set a to $[a_p, a_{p-1}, \ldots, a_1, a_{p+1}, a_{p+2}, \ldots, a_n]$.

Find a way to sort a using no more than $\frac{5n}{2}$ reversals of the above kind, or determine that such a way doesn't exist. The number of reversals doesn't have to be minimized.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains a single integer n ($3 \le n \le 2021$; n is odd) — the length of the permutation.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the permutation itself.

It is guaranteed that the sum of n over all test cases does not exceed 2021.

Output

For each test case, if it's impossible to sort the given permutation in at most $\frac{5n}{2}$ reversals, print a single integer -1.

Otherwise, print an integer m ($0 \le m \le \frac{5n}{2}$), denoting the number of reversals in your sequence of steps, followed by m integers p_i ($1 \le p_i \le n$; p_i is odd), denoting the lengths of the prefixes of a to be reversed, in chronological order.

Note that m doesn't have to be minimized. If there are multiple answers, print any.

Example

input

```
3
3
1 2 3
5
3 4 5 2 1
3
2 1 3
```

output

```
4
3 3 3 3
2
3 5
-1
```

Note

In the first test case, the permutation is already sorted. Any even number of reversals of the length 3 prefix doesn't change that fact.

In the second test case, after reversing the prefix of length 3 the permutation will change to $[5, 4, 3, 2, 1]$, and then after reversing the prefix of length 5 the permutation will change to $[1, 2, 3, 4, 5]$.

In the third test case, it's impossible to sort the permutation.

D. Top-Notch Insertions

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Consider the insertion sort algorithm used to sort an integer sequence $[a_1, a_2, \ldots, a_n]$ of length n in non-decreasing order.

For each i in order from 2 to n, do the following. If $a_i \ge a_{i-1}$, do nothing and move on to the next value of i. Otherwise, find the smallest j such that $a_i < a_j$, shift the elements on positions from j to $i-1$ by one position to the right, and write down the initial value of a_i to position j. In this case we'll say that we performed an *insertion* of an element from position i to position j.

It can be noticed that after processing any i, the prefix of the sequence $[a_1, a_2, \ldots, a_i]$ is sorted in non-decreasing order, therefore, the algorithm indeed sorts any sequence.

For example, sorting $[4, 5, 3, 1, 3]$ proceeds as follows:

- $i = 2$: $a_2 \ge a_1$, do nothing;
- $i = 3$: $j = 1$, insert from position 3 to position 1: $[3, 4, 5, 1, 3]$;
- $i = 4$: $j = 1$, insert from position 4 to position 1: $[1, 3, 4, 5, 3]$;
- $i = 5$: $j = 3$, insert from position 5 to position 3: $[1, 3, 3, 4, 5]$.

You are given an integer n and a list of m integer pairs (x_i, y_i). We are interested in sequences such that if you sort them using the above algorithm, exactly m insertions will be performed: first from position x_1 to position y_1, then from position x_2 to position y_2, ..., finally, from position x_m to position y_m.

How many sequences of length n consisting of (not necessarily distinct) integers between 1 and n, inclusive, satisfy the above condition? Print this number modulo $998\,244\,353$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \le n \le 2 \cdot 10^5$; $0 \le m < n$) — the length of the sequence and the number of insertions.

The i-th of the following m lines contains two integers x_i and y_i ($2 \le x_1 < x_2 < \ldots < x_m \le n$; $1 \le y_i < x_i$). These lines describe the sequence of insertions in chronological order.

It is guaranteed that the sum of m over all test cases does not exceed $2 \cdot 10^5$. **Note that there is no constraint on the sum of n of the same kind.**

Output

For each test case, print the number of sequences of length n consisting of integers from 1 to n such that sorting them with the described algorithm produces the given sequence of insertions, modulo $998\,244\,353$.

Example

input

```
3
3 0
3 2
2 1
3 1
5 3
3 1
4 1
5 3
```

output

```
10
1
21
```

Note

In the first test case, the algorithm performs no insertions — therefore, the initial sequence is already sorted in non-decreasing order. There are 10 such sequences: $[1,1,1]$, $[1,1,2]$, $[1,1,3]$, $[1,2,2]$, $[1,2,3]$, $[1,3,3]$, $[2,2,2]$, $[2,2,3]$, $[2,3,3]$, $[3,3,3]$.

In the second test case, the only sequence satisfying the conditions is $[3,2,1]$.

In the third test case, $[4,5,3,1,3]$ is one of the sought sequences.

E. Down Below

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

In a certain video game, the player controls a hero characterized by a single integer value: *power*.

On the current level, the hero got into a system of n caves numbered from 1 to n, and m tunnels between them. Each tunnel connects two distinct caves. Any two caves are connected with at most one tunnel. Any cave can be reached from any other cave by moving via tunnels.

The hero starts the level in cave 1, and every other cave contains a monster.

The hero can move between caves via tunnels. If the hero leaves a cave and enters a tunnel, he must finish his movement and arrive at the opposite end of the tunnel.

The hero can use each tunnel to move in both directions. However, the hero **can not** use the same tunnel **twice in a row**. Formally, if the hero has just moved from cave i to cave j via a tunnel, he can not head back to cave i immediately after, but he can head to any other cave connected to cave j with a tunnel.

It is known that at least two tunnels come out of every cave, thus, the hero will never find himself in a dead end even considering the above requirement.

To pass the level, the hero must beat the monsters in all the caves. When the hero enters a cave for the first time, he will have to fight the monster in it. The hero can beat the monster in cave i if and only if the hero's power is strictly greater than a_i. In case of beating the monster, the hero's power increases by b_i. If the hero can't beat the monster he's fighting, the game ends and the player loses.

After the hero beats the monster in cave i, all subsequent visits to cave i won't have any consequences: the cave won't have any monsters, and the hero's power won't change either.

Find the smallest possible power the hero must start the level with to be able to beat all the monsters and pass the level.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of each test case contains two integers n and m ($3 \le n \le 1000$; $n \le m \le min(\frac{n(n-1)}{2}, 2000)$) — the number of caves and tunnels.

The second line contains $n-1$ integers $a_2, a_3, \dots, a_n$ ($1 \le a_i \le 10^9$) — values the hero's power are compared to while fighting monsters in caves $2, 3, \dots, n$.

The third line contains $n-1$ integers $b_2, b_3, \dots, b_n$ ($1 \le b_i \le 10^9$) — increases applied to the hero's power for beating monsters in caves $2, 3, \dots, n$.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$) — the numbers of caves connected with a tunnel.

No two caves are connected with more than one tunnel. Any cave can be reached from any other cave by moving via tunnels. At least two tunnels come out of every cave.

It is guaranteed that the sum of n over all test cases does not exceed 1000, and the sum of m over all test cases does not exceed 2000.

Output

For each test case print a single integer — the smallest possible power the hero must start the level with to be able to beat all the monsters and pass the level.

Example

input

```
3
4 4
11 22 13
8 7 5
1 2
2 3
3 4
4 1
4 4
11 22 13
5 7 8
1 2
2 3
3 4
4 1
5 7
10 40 20 30
7 2 10 5
1 2
1 5
2 3
2 4
2 5
3 4
4 5
```

output

```
15
15
19
```

Note

In the first test case, the hero can pass the level with initial power 15 as follows:

- move from cave 1 to cave 2: since $15 > 11$, the hero beats the monster, and his power increases to $15 + 8 = 23$;
- move from cave 2 to cave 3: since $23 > 22$, the hero beats the monster, and his power increases to $23 + 7 = 30$;
- move from cave 3 to cave 4: since $30 > 13$, the hero beats the monster, and his power increases to $30 + 5 = 35$.

In the second test case, the situation is similar except that the power increases for beating monsters in caves 2 and 4 are exchanged. The hero can follow a different route, $1 \to 4 \to 3 \to 2$, and pass the level with initial power 15.

In the third test case, the hero can pass the level with initial power 19 as follows:

- move from cave 1 to cave 2: since $19 > 10$, the hero beats the monster, and his power increases to $19 + 7 = 26$;
- move from cave 2 to cave 4: since $26 > 20$, the hero beats the monster, and his power increases to $26 + 10 = 36$;
- move from cave 4 to cave 5: since $36 > 30$, the hero beats the monster, and his power increases to $36 + 5 = 41$;
- move from cave 5 to cave 2: there is no monster in this cave anymore, nothing happens;
- move from cave 2 to cave 3: since $41 > 40$, the hero beats the monster, and his power increases to $41 + 2 = 43$.

F. Strange Sort

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You have a permutation: an array $a = [a_1, a_2, \dots, a_n]$ of distinct integers from 1 to n. The length of the permutation n is odd.

Consider the following algorithm of sorting the permutation in increasing order.

A helper procedure of the algorithm, $f(i)$, takes a single argument i ($1 \le i \le n-1$) and does the following. If $a_i > a_{i+1}$, the values of a_i and a_{i+1} are exchanged. Otherwise, the permutation doesn't change.

The algorithm consists of iterations, numbered with consecutive integers starting with 1. On the i-th iteration, the algorithm does the following:

- if i is odd, call $f(1), f(3), \ldots, f(n-2)$;
- if i is even, call $f(2), f(4), \ldots, f(n-1)$.

It can be proven that after a finite number of iterations the permutation will be sorted in increasing order.

After how many iterations will this happen for the first time?

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 2 \cdot 10^5 - 1$; n is odd) — the length of the permutation.

The second line contains n distinct integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the permutation itself.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5 - 1$.

Output

For each test case print the number of iterations after which the permutation will become sorted in increasing order for the first time.

If the given permutation is already sorted, print 0.

Example

input

```
3
3
3 2 1
7
4 5 7 1 3 2 6
5
1 2 3 4 5
```

output

```
3
5
0
```

Note

In the first test case, the permutation will be changing as follows:

- after the 1-st iteration: $[2, 3, 1]$;
- after the 2-nd iteration: $[2, 1, 3]$;
- after the 3-rd iteration: $[1, 2, 3]$.

In the second test case, the permutation will be changing as follows:

- after the 1-st iteration: $[4, 5, 1, 7, 2, 3, 6]$;
- after the 2-nd iteration: $[4, 1, 5, 2, 7, 3, 6]$;
- after the 3-rd iteration: $[1, 4, 2, 5, 3, 7, 6]$;
- after the 4-th iteration: $[1, 2, 4, 3, 5, 6, 7]$;
- after the 5-th iteration: $[1, 2, 3, 4, 5, 6, 7]$.

In the third test case, the permutation is already sorted and the answer is 0.