A. Distinct Buttons

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Deemo - Entrance

You are located at the point $(0, 0)$ of an infinite Cartesian plane. You have a controller with 4 buttons which can perform one of the following operations:

- `U`: move from (x, y) to $(x, y + 1)$;
- `R`: move from (x, y) to $(x + 1, y)$;
- `D`: move from (x, y) to $(x, y - 1)$;
- `L`: move from (x, y) to $(x - 1, y)$.

Unfortunately, the controller is broken. If you press all the 4 buttons (in any order), the controller stops working. It means that, during the whole trip, you can only press at most 3 distinct buttons (any number of times, in any order).

There are n special points in the plane, with integer coordinates (x_i, y_i).

Can you visit all the special points (in any order) without breaking the controller?

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 100$) — the number of special points.

Each of the next n lines contains two integers x_i, y_i ($-100 \le x_i, y_i \le 100$), which represent the special point (x_i, y_i).

Note that there are no constraints on the sum of n over all test cases.

Output

For each test case, output "YES" (without quotes), if you can reach all the special points without breaking the controller, and "NO" (without quotes) otherwise.

You may output each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as positive answer).

Standard Input	Standard Output
6 3 1 -1 0 0 1 -1 4 -3 -2 -3 -1 -3 0	YES YES NO NO YES YES

-3 1 4 1 1 -1 -1 1 -1 -1 1 6 -4 14 -9 -13 -14 5 14 15 -8 -4 19 9 6 82 64 39 91 3 46 87 83 74 21 7 25 1 100 -100	

Note

In the first test case, you can move as follows:

- you start from $(0, 0)$;
- you visit the special point $(x_2, y_2) = (0, 0)$;
- you press R, and you move from $(0, 0)$ to $(1, 0)$;
- you press D, and you move from $(1, 0)$ to $(1, -1)$;
- you visit the special point $(x_1, y_1) = (1, -1)$;
- you visit the special point $(x_3, y_3) = (1, -1)$.

Therefore, you can visit all the special points using only the buttons R, D, so the controller does not break.

Note that the special points may coincide.

In the second test case, you can show that you can visit all the special points using only the buttons U, D, L.

In the third test case, you can show that you must press all the buttons (U, R, D, L) to visit all the points, so the controller would break.

,

B. Make Almost Equal With Mod

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

xi - Solar Storm

You are given an array $a_1, a_2, \ldots, a_n$ of distinct positive integers. You have to do the following operation **exactly once**:

- choose a positive integer k;
- for each i from 1 to n, replace a_i with $a_i \bmod k$[†].

Find a value of k such that $1 \le k \le 10^{18}$ and the array $a_1, a_2, \ldots, a_n$ contains **exactly** 2 distinct values at the end of the operation. It can be shown that, under the constraints of the problem, at least one such k always exists. If there are multiple solutions, you can print any of them.

[†] $a \bmod b$ denotes the remainder after dividing a by b. For example:

- $7 \bmod 3 = 1$ since $7 = 3 \cdot 2 + 1$
- $15 \bmod 4 = 3$ since $15 = 4 \cdot 3 + 3$
- $21 \bmod 1 = 0$ since $21 = 21 \cdot 1 + 0$

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 first line of each test case contains a single integer n ($2 \le n \le 100$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{17}$) — the initial state of the array. It is guaranteed that all the a_i are distinct.

Note that there are no constraints on the sum of n over all test cases.

Output

For each test case, output a single integer: a value of k ($1 \le k \le 10^{18}$) such that the array $a_1, a_2, \ldots, a_n$ contains exactly 2 distinct values at the end of the operation.

Standard Input	Standard Output
5 4 8 15 22 30 5 60 90 98 120 308 6 328 769 541 986 215 734 5 1000 2000 7000 11000 16000 2	7 30 3 5000 1000000000000000000

2 1	

Note

In the first test case, you can choose $k = 7$. The array becomes $[8 \bmod 7, 15 \bmod 7, 22 \bmod 7, 30 \bmod 7] = [1, 1, 1, 2]$, which contains exactly 2 distinct values ($\{1, 2\}$).

In the second test case, you can choose $k = 30$. The array becomes $[0, 0, 8, 0, 8]$, which contains exactly 2 distinct values ($\{0, 8\}$). Note that choosing $k = 10$ would also be a valid solution.

In the last test case, you can choose $k = 10^{18}$. The array becomes $[2, 1]$, which contains exactly 2 distinct values ($\{1, 2\}$). Note that choosing $k = 10^{18} + 1$ would not be valid, because $1 \leq k \leq 10^{18}$ must be true.

,

C. Heavy Intervals

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Shiki - Pure Ruby

You have n intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$, such that $l_i < r_i$ for each i, and all the endpoints of the intervals are distinct.

The i-th interval has weight c_i per unit length. Therefore, the *weight* of the i-th interval is $c_i \cdot (r_i - l_i)$.

You don't like large weights, so you want to make the sum of weights of the intervals as small as possible. It turns out you can perform all the following three operations:

- rearrange the elements in the array l in any order;
- rearrange the elements in the array r in any order;
- rearrange the elements in the array c in any order.

However, after performing all of the operations, the intervals must still be valid (i.e., for each i, $l_i < r_i$ must hold).

What's the minimum possible sum of weights of the intervals after performing the operations?

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 ($1 \le n \le 10^5$) — the number of intervals.

The second line of each test case contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le 2 \cdot 10^5$) — the left endpoints of the initial intervals.

The third line of each test case contains n integers $r_1, r_2, \ldots, r_n$ ($l_i < r_i \le 2 \cdot 10^5$) — the right endpoints of the initial intervals.

It is guaranteed that $\{l_1, l_2, \ldots, l_n, r_1, r_2, \ldots, r_n\}$ are all distinct.

The fourth line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^7$) — the initial weights of the intervals per unit length.

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

Output

For each test case, output a single integer: the minimum possible sum of weights of the intervals after your operations.

Standard Input	Standard Output
2 2 8 3	2400 42

12 23 100 100 4 20 1 2 5 30 4 3 10 2 3 2 3	

Note

In the first test case, you can make

- $l = [8, 3]$;
- $r = [23, 12]$;
- $c = [100, 100]$.

In that case, there are two intervals:

- interval $[8, 23]$ with weight 100 per unit length, and $100 \cdot (23 - 8) = 1500$ in total;
- interval $[3, 12]$ with weight 100 per unit length, and $100 \cdot (12 - 3) = 900$ in total.

The sum of the weights is 2400. It can be shown that there is no configuration of final intervals whose sum of weights is less than 2400.

In the second test case, you can make

- $l = [1, 2, 5, 20]$;
- $r = [3, 4, 10, 30]$;
- $c = [3, 3, 2, 2]$.

In that case, there are four intervals:

- interval $[1, 3]$ with weight 3 per unit length, and $3 \cdot (3 - 1) = 6$ in total;
- interval $[2, 4]$ with weight 3 per unit length, and $3 \cdot (4 - 2) = 6$ in total;
- interval $[5, 10]$ with weight 2 per unit length, and $2 \cdot (10 - 5) = 10$ in total;
- interval $[20, 30]$ with weight 2 per unit length, and $2 \cdot (30 - 20) = 20$ in total.

The sum of the weights is 42. It can be shown that there is no configuration of final intervals whose sum of weights is less than 42.

,

D. Split Plus K

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

eliteLAQ - Desert Ruins

There are n positive integers $a_1, a_2, \ldots, a_n$ on a blackboard. You are also given a positive integer k. You can perform the following operation some (possibly 0) times:

- choose a number x on the blackboard;
- erase one occurrence of x;
- write two **positive** integers y, z such that $y + z = x + k$ on the blackboard.

Is it possible to make all the numbers on the blackboard equal? If yes, what is the minimum number of operations you need?

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 ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le 10^{12}$) — the number of integers initially on the blackboard and the constant k.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{12}$) — the initial state of the blackboard.

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 line containing an integer: the minimum number of operations you need to make all the numbers on the blackboard equal, or -1 if it is impossible.

Standard Input	Standard Output
9 2 1 3 4 2 3 7 11 3 10 100 40 100 2 1 1 2 2 2 1 2 1 327869541 327869541 5 26250314066	3 1 4 -1 -1 0 3119 28999960732 -1

439986238782 581370817372 409476934981 287439719777 737637983182 5 616753575719 321037808624 222034505841 214063039282 441536506916 464097941819 5 431813672576 393004301966 405902283416 900951084746 672201172466 518769038906	

Note

In the first test case, $k = 1$. You can make all the numbers on the blackboard equal to 2 with the following operations:

- Erase $x = 4$ and write $(y, z) = (2, 3)$. Note that $y + z = x + k$. The blackboard now contains the multiset $\{3, 2, 3\}$.
- Erase $x = 3$ and write $(y, z) = (2, 2)$. Note that $y + z = x + k$. The blackboard now contains $\{2, 2, 2, 3\}$.
- Erase $x = 3$ and write $(y, z) = (2, 2)$. Note that $y + z = x + k$. The blackboard now contains $\{2, 2, 2, 2, 2\}$.

This makes all the numbers equal in 3 operations. It can be shown that you cannot make all the numbers equal in less than 3 operations.

In the second test case, $k = 3$. You can make all the numbers on the blackboard equal to 7 with the following operation:

- Erase $x = 11$ and write $(y, z) = (7, 7)$. Note that $y + z = x + k$. The blackboard now contains $\{7, 7, 7\}$.

In the third test case, $k = 10$. You can make all the numbers on the blackboard equal to 40 with the following operations:

- Erase $x = 100$ and write $(y, z) = (70, 40)$. Note that $y + z = x + k$. The blackboard now contains $\{70, 40, 40, 100\}$.
- Erase $x = 70$ and write $(y, z) = (40, 40)$. Note that $y + z = x + k$. The blackboard now contains $\{40, 40, 40, 40, 100\}$.
- Erase $x = 100$ and write $(y, z) = (40, 70)$. Note that $y + z = x + k$. The blackboard now contains $\{40, 40, 40, 40, 40, 70\}$.
- Erase $x = 70$ and write $(y, z) = (40, 40)$. Note that $y + z = x + k$. The blackboard now contains $\{40, 40, 40, 40, 40, 40, 40\}$.

In the fourth and in the fifth test case, you can show that it is impossible to make all the numbers on the blackboard equal.

,

E. Multiple Lamps

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

Kid2Will - Fire Aura

You have n lamps, numbered from 1 to n. Initially, all the lamps are turned off.

You also have n buttons. The i-th button toggles all the lamps whose index is a multiple of i. When a lamp is toggled, if it was off it turns on, and if it was on it turns off.

You have to press some buttons according to the following rules.

- You have to press at least one button.
- You cannot press the same button multiple times.
- You are given m pairs (u_i, v_i). If you press the button u_i, you also have to press the button v_i (at any moment, not necessarily after pressing the button u_i). Note that, if you press the button v_i, you don't need to press the button u_i.

You don't want to waste too much electricity. Find a way to press buttons such that at the end at most $\lfloor n/5 \rfloor$ lamps are on, or print -1 if it is impossible.

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 and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the number of lamps and the number of pairs, respectively.

Each of the next m lines contains two integers u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). If you press the button u_i, you also have to press the button v_i. It is guaranteed that the pairs (u_i, v_i) are distinct.

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

Output

For each test case:

- If there is no choice of buttons that makes at most $\lfloor n/5 \rfloor$ lamps on at the end, output a single line containing -1.
- Otherwise, output two lines. The first line should contain an integer k ($1 \le k \le n$) — the number of pressed buttons. The second line should contain k integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le n$) — the indices of the pressed buttons (in any order). The b_i must be distinct, and at the end at most $\lfloor n/5 \rfloor$ lamps must be turned on.

Standard Input	Standard Output
4 4 0 5 2 4 1	-1 4 3 5 1 2 3

5 1 15 9 7 8 8 9 9 10 10 9 11 1 12 2 13 3 14 4 15 5 5 4 1 2 2 3 3 4 4 5	8 9 10 1 5

Note

In the first test case, you need to turn at most $\lfloor 4/5 \rfloor$ lamps on, which means that no lamp can be turned on. You can show that no choice of at least one button turns 0 lamps on.

In the second test case, you can press buttons 3, 5, 1, 2.

- Initially, all the lamps are off;
- after pressing button 3, the lamps whose index is a multiple of 3 (i.e., 3) are toggled, so lamp 3 is turned on;
- after pressing button 5, the lamps whose index is a multiple of 5 (i.e., 5) are toggled, so lamps 3, 5 are turned on;
- after pressing button 1, the lamps whose index is a multiple of 1 (i.e., 1, 2, 3, 4, 5) are toggled, so lamps 1, 2, 4 are turned on;
- after pressing button 2, the lamps whose index is a multiple of 2 (i.e., 2, 4) are toggled, so lamp 1 is turned on.

This is valid because

- you pressed at least one button;
- you pressed all the buttons at most once;
- you pressed button $u_2 = 5$, which means that you had to also press button $v_2 = 1$: in fact, button 1 has been pressed;
- at the end, only lamp 1 is on.

In the third test case, pressing the buttons 8, 9, 10 turns only the lamps 8, 9, 10 on.

,

F1. Small Permutation Problem (Easy Version)

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

Andy Tunstall - MiniBoss

In the easy version, the a_i are in the range $[0, n]$; in the hard version, the a_i are in the range $[-1, n]$ and the definition of good permutation is slightly different. You can make hacks only if all versions of the problem are solved.

You are given an integer n and an array $a_1, a_2 \ldots, a_n$ of integers in the range $[0, n]$.

A permutation $p_1, p_2, \ldots, p_n$ of $[1, 2, \ldots, n]$ is good if, for each i, the following condition is true:

- the number of values $\leq i$ in $[p_1, p_2, \ldots, p_i]$ is exactly a_i.

Count the good permutations of $[1, 2, \ldots, n]$, modulo $998\,244\,353$.

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 a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq n$), which describe the conditions for a good permutation.

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 line containing the number of good permutations, modulo $998\,244\,353$.

Standard Input	Standard Output
5 5 1 2 3 4 5 6 0 2 2 2 4 6 6 0 1 3 4 5 5 6 1 2 3 2 4 6 15 0 0 1 1 1 2 3 4 5 6 7 9 11 13 15	1 4 0 0 532305727

Note

In the first test case, the only good permutation is $[1, 2, 3, 4, 5]$.

In the second test case, there are 4 good permutations: $[2, 1, 5, 6, 3, 4]$, $[2, 1, 5, 6, 4, 3]$, $[2, 1, 6, 5, 3, 4]$, $[2, 1, 6, 5, 4, 3]$. For example, $[2, 1, 5, 6, 3, 4]$ is good because:

- $a_1 = 0$, and there are 0 values ≤ 1 in $[p_1] = [2]$;
- $a_2 = 2$, and there are 2 values ≤ 2 in $[p_1, p_2] = [2, 1]$;
- $a_3 = 2$, and there are 2 values ≤ 3 in $[p_1, p_2, p_3] = [2, 1, 5]$;
- $a_4 = 2$, and there are 2 values ≤ 4 in $[p_1, p_2, p_3, p_4] = [2, 1, 5, 6]$;
- $a_5 = 4$, and there are 4 values ≤ 5 in $[p_1, p_2, p_3, p_4, p_5] = [2, 1, 5, 6, 3]$;
- $a_6 = 6$, and there are 6 values ≤ 6 in $[p_1, p_2, p_3, p_4, p_5, p_6] = [2, 1, 5, 6, 3, 4]$.

In the third test case, there are no good permutations, because there are no permutations with $a_6 = 5$ values ≤ 6 in $[p_1, p_2, p_3, p_4, p_5, p_6]$.

,

F2. Small Permutation Problem (Hard Version)

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

Andy Tunstall - MiniBoss

In the easy version, the a_i are in the range $[0, n]$; in the hard version, the a_i are in the range $[-1, n]$ and the definition of good permutation is slightly different. You can make hacks only if all versions of the problem are solved.

You are given an integer n and an array $a_1, a_2, \ldots, a_n$ of integers in the range $[-1, n]$.

A permutation $p_1, p_2, \ldots, p_n$ of $[1, 2, \ldots, n]$ is good if, for each i, the following condition is true:

- if $a_i \neq -1$, the number of values $\leq i$ in $[p_1, p_2, \ldots, p_i]$ is exactly a_i.

Count the good permutations of $[1, 2, \ldots, n]$, modulo $998\,244\,353$.

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 a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \leq a_i \leq n$), which describe the conditions for a good permutation.

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 line containing the number of good permutations, modulo $998\,244\,353$.

Standard Input	Standard Output
10 5 -1 -1 -1 -1 -1 5 1 2 3 4 5 6 0 2 2 2 -1 -1 6 -1 -1 -1 -1 -1 5 6 -1 -1 3 2 -1 -1 15 0 0 -1 -1 -1 2 2 -1 -1 -1 -1 9 11 13 15 6 0 2 2 2 4 6 6	120 1 4 0 0 494403526 4 0 0 532305727

0 1 3 4 5 5 6 1 2 3 2 4 6 15 0 0 1 1 1 2 3 4 5 6 7 9 11 13 15	

Note

In the first test case, all the permutations of length 5 are good, so there are 120 good permutations.

In the second test case, the only good permutation is $[1, 2, 3, 4, 5]$.

In the third test case, there are 4 good permutations: $[2, 1, 5, 6, 3, 4]$, $[2, 1, 5, 6, 4, 3]$, $[2, 1, 6, 5, 3, 4]$, $[2, 1, 6, 5, 4, 3]$. For example, $[2, 1, 5, 6, 3, 4]$ is good because:

- $a_1 = 0$, and there are 0 values ≤ 1 in $[p_1] = [2]$;
- $a_2 = 2$, and there are 2 values ≤ 2 in $[p_1, p_2] = [2, 1]$;
- $a_3 = 2$, and there are 2 values ≤ 3 in $[p_1, p_2, p_3] = [2, 1, 5]$;
- $a_4 = 2$, and there are 2 values ≤ 4 in $[p_1, p_2, p_3, p_4] = [2, 1, 5, 6]$;
- $a_5 = -1$, so there are no restrictions on $[p_1, p_2, p_3, p_4, p_5]$;
- $a_6 = -1$, so there are no restrictions on $[p_1, p_2, p_3, p_4, p_5, p_6]$.

,

G. Pumping Lemma

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

Tanchiky & Siromaru - Crystal Gravity

You are given two strings s, t of length n, m, respectively. Both strings consist of lowercase letters of the English alphabet.

Count the triples (x, y, z) of strings such that the following conditions are true:

- $s = x + y + z$ (the symbol $+$ represents the concatenation);
- $t = x + \underbrace{y + \cdots + y}_{k \text{ times}} + z$ for some integer k.

Input

The first line contains two integers n and m ($1 \le n < m \le 10^7$) — the length of the strings s and t, respectively.

The second line contains the string s of length n, consisting of lowercase letters of the English alphabet.

The third line contains the string t of length m, consisting of lowercase letters of the English alphabet.

Output

Output a single integer: the number of valid triples (x, y, z).

Standard Input	Standard Output
4 8 abcd abcbcbcd	1
3 5 aaa aaaaa	5
12 16 abbababacaab abbababababacaab	8

Note

In the first test case, the only valid triple is $(x, y, z) = (\texttt{"a"}, \texttt{"bc"}, \texttt{"d"})$. In fact,

- $\texttt{"abcd"} = \texttt{"a"} + \texttt{"bc"} + \texttt{"d"}$;
- $\texttt{"abcbcbcd"} = \texttt{"a"} + \texttt{"bc"} + \texttt{"bc"} + \texttt{"bc"} + \texttt{"d"}$.

In the second test case, there are 5 valid triples:

- $(x, y, z) = (\texttt{""}, \texttt{"a"}, \texttt{"aa"})$;
- $(x, y, z) = (\texttt{""}, \texttt{"aa"}, \texttt{"a"})$;
- $(x, y, z) = (\texttt{"a"}, \texttt{"a"}, \texttt{"a"})$;
- $(x, y, z) = (\texttt{"a"}, \texttt{"aa"}, \texttt{""})$;
- $(x, y, z) = (\texttt{"aa"}, \texttt{"a"}, \texttt{""})$.

In the third test case, there are 8 valid triples:

- $(x, y, z) = (\texttt{"ab"}, \texttt{"ba"}, \texttt{"babacaab"})$;
- $(x, y, z) = (\texttt{"abb"}, \texttt{"ab"}, \texttt{"abacaab"})$;
- $(x, y, z) = (\texttt{"abba"}, \texttt{"ba"}, \texttt{"bacaab"})$;
- $(x, y, z) = (\texttt{"ab"}, \texttt{"baba"}, \texttt{"bacaab"})$;
- $(x, y, z) = (\texttt{"abbab"}, \texttt{"ab"}, \texttt{"acaab"})$;
- $(x, y, z) = (\texttt{"abb"}, \texttt{"abab"}, \texttt{"acaab"})$;
- $(x, y, z) = (\texttt{"abbaba"}, \texttt{"ba"}, \texttt{"caab"})$;
- $(x, y, z) = (\texttt{"abba"}, \texttt{"baba"}, \texttt{"caab"})$.

,

H. Parallel Swaps Sort

Input file:	standard input
Output file:	standard output
Time limit:	7 seconds
Memory limit:	1024 megabytes

Dubmood - Keygen 8

You are given a permutation $p_1, p_2, \ldots, p_n$ of $[1, 2, \ldots, n]$. You can perform the following operation some (possibly 0) times:

- choose a subarray $[l, r]$ of even length;
- swap a_l, a_{l+1};
- swap a_{l+2}, a_{l+3} (if $l + 3 \leq r$);
- $\ldots$
- swap a_{r-1}, a_r.

Sort the permutation in at most 10^6 operations. You do not need to minimize the number of operations.

Input

The first line contains a single integer n ($2 \leq n \leq 3 \cdot 10^5$) — the length of the permutation.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$, the p_i are distinct) — the permutation before performing the operations.

Output

Output your operations in the following format.

The first line should contain an integer k ($0 \leq k \leq 10^6$) — the number of operations.

The next k lines represent the k operations in order. Each of these k lines should contain two integers l and r ($1 \leq l < r \leq n$, $r - l + 1$ must be even) — the corresponding operation consists in choosing the subarray $[l, r]$ and swapping its elements according to the problem statement.

After all the operations, $a_i = i$ must be true for each i ($1 \leq i \leq n$).

Standard Input	Standard Output
5 2 5 4 1 3	5 1 4 1 2 2 5 1 4 4 5
9 1 2 3 4 5 6 7 8 9	0
10 6 4 2 3 8 10 9 1 5 7	15 1 8 6 9 1 8

	3 10 1 10 1 10 1 6 6 9 6 9 2 7 9 10 5 10 1 6 2 9 1 10

Note

In the first test:

- At the beginning, $p = [2, 5, 4, 1, 3]$.
- In the first operation, you can choose $[l, r] = [1, 4]$. Then, (a_1, a_2) are swapped and (a_3, a_4) are swapped. The new permutation is $p = [5, 2, 1, 4, 3]$.
- In the second operation, you can choose $[l, r] = [1, 2]$. Then, (a_1, a_2) are swapped. The new permutation is $p = [2, 5, 1, 4, 3]$.
- In the third operation, you can choose $[l, r] = [2, 5]$. Then, (a_2, a_3) are swapped and (a_4, a_5) are swapped. The new permutation is $p = [2, 1, 5, 3, 4]$.
- In the fourth operation, you can choose $[l, r] = [1, 4]$. Then, (a_1, a_2) are swapped and (a_3, a_4) are swapped. The new permutation is $p = [1, 2, 3, 5, 4]$.
- In the fifth operation, you can choose $[l, r] = [4, 5]$. Then, (a_4, a_5) are swapped. The new permutation is $p = [1, 2, 3, 4, 5]$, which is sorted.

In the second test, the permutation is already sorted, so you do not need to perform any operation.

,

I. Short Permutation Problem

```
Input file:     standard input
Output file:    standard output
Time limit:     7 seconds
Memory limit:   1024 megabytes
```

Xomu - Last Dance

You are given an integer n.

For each (m, k) such that $3 \le m \le n+1$ and $0 \le k \le n-1$, count the permutations of $[1, 2, \ldots, n]$ such that $p_i + p_{i+1} \ge m$ for exactly k indices i, modulo $998\,244\,353$.

Input

The input consists of a single line, which contains two integers n, x ($2 \le n \le 4000$, $1 \le x < 1\,000\,000\,007$).

Output

Let $a_{m,k}$ be the answer for the pair (m, k), modulo $998\,244\,353$.

Let

$$S = \sum_{m=3}^{n+1} \sum_{k=0}^{n-1} a_{m,k} x^{mn+k} \quad .$$

Output a single line with an integer: S modulo $1\,000\,000\,007$.

Note that using two different modulos is intentional. We want you to calculate all the $a_{m,k}$ modulo $998\,244\,353$, then treat them like integers in the range $[0, 998\,244\,352]$, and hash them modulo $1\,000\,000\,007$.

Standard Input	Standard Output
3 2	77824
4 1000000000	30984329
8 327869541	85039220
4000 1149333	584870166

Note

In the first test case, the answers for all (m, k) are shown in the following table:

	$k=0$	$k=1$	$k=2$
$m=3$	0	0	6
$m=4$	0	4	2

- The answer for $(m, k) = (3, 2)$ is 6, because for every permutation of length 3, $a_i + a_{i+1} \ge 3$ exactly 2 times.

- The answer for $(m, k) = (4, 2)$ is 2. In fact, there are 2 permutations of length 3 such that $a_i + a_{i+1} \geq 4$ exactly 2 times: $[1, 3, 2]$, $[2, 3, 1]$.

Therefore, the value to print is
$2^9 \cdot 0 + 2^{10} \cdot 0 + 2^{11} \cdot 6 + 2^{12} \cdot 0 + 2^{13} \cdot 4 + 2^{14} \cdot 2 \equiv 77\,824 \pmod{1\,000\,000\,007}$.

In the second test case, the answers for all (m, k) are shown in the following table:

	$k = 0$	$k = 1$	$k = 2$	$k = 3$
$m = 3$	0	0	0	24
$m = 4$	0	0	12	12
$m = 5$	0	4	16	4

- The answer for $(m, k) = (5, 1)$ is 4. In fact, there are 4 permutations of length 4 such that $a_i + a_{i+1} \geq 5$ exactly 1 time: $[2, 1, 3, 4]$, $[3, 1, 2, 4]$, $[4, 2, 1, 3]$, $[4, 3, 1, 2]$.

In the third test case, the answers for all (m, k) are shown in the following table:

	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
$m = 3$	0	0	0	0	0	0	0	40320
$m = 4$	0	0	0	0	0	0	10080	30240
$m = 5$	0	0	0	0	0	1440	17280	21600
$m = 6$	0	0	0	0	480	8640	21600	9600
$m = 7$	0	0	0	96	3456	16416	16896	3456
$m = 8$	0	0	48	2160	12960	18240	6480	432
$m = 9$	0	16	1152	9648	18688	9648	1152	16