Problem A. Attack Order

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

In a certain game, you control a board of n minions numbered from 1 to n. Each minion i is characterized by an integer a_i, called its *attack*.

For the upcoming fight, you will arrange the minions in a line from left to right.

After that, some of the minions' attacks will get *buffed*. The ability of each minion i reads "Before the fight, increase the attack of *another* random minion by b_i". Formally, for each i, an arbitrary minion $j \neq i$ will be chosen and its attack a_j will be increased by b_i.

Note that the buff choices are independent and happen simultaneously. In particular, the attack of any minion can get buffed multiple times.

You want the attacks of the minions to be *non-increasing* from left to right after all the buffs happen. Determine whether it's possible for you to arrange the minions in a way that guarantees that, regardless of buff choices.

Input

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

The first line of each test case contains a single integer n ($2 \le n \le 100$).

The i-th of the next n lines contains two integers a_i and b_i ($0 \le a_i, b_i \le 10^6$).

Output

For each test case, print "`Yes`" if it is possible to arrange the minions in such a way that their attacks will be non-increasing regardless of buff choices, and "`No`" otherwise.

Example

standard input	*standard output*
3 2 15 25 10 5 3 7 0 7 3 10 0 3 10 10 20 20 30 30	Yes Yes No

Note

In the first example test case, the minions buff each other. The attacks of minions 1 and 2 during the fight will always be 20 and 35, respectively. You can arrange the minions in order $\langle 2, 1 \rangle$.

In the second example test case, only minion 2 buffs someone else. One valid ordering is $\langle 3, 1, 2 \rangle$. If minion 2 buffs minion 1, the attacks of the minions, from left to right, will be $\langle 10, 10, 7 \rangle$. If minion 2 buffs minion 3, the attacks of the minions will be $\langle 13, 7, 7 \rangle$. Both sequences are non-increasing.

Problem B. Browsing the Collection

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

You are browsing an online collection of n items numbered from 1 to n arranged on a circle. The item to the right of each item i is item $i+1$, and the item to the right of item n is item 1. Similarly, the item to the left of each item i is item $i-1$, and the item to the left of item 1 is item n.

The items have m *parameters* numbered from 1 to m. The value of parameter j for item i is an integer $a_{i,j}$.

While you are browsing, at any moment, there is a pointer directed at some item, called the *current* item.

Moreover, you can manage a set of *filtering conditions*. Each condition is a pair (j, v), meaning that the j-th parameter of the item must be equal to v. The current item always satisfies all conditions in the set.

To browse through the collection, you can do *operations*. Each operation must have one of the following four kinds:

- Click "right". The pointer moves to the closest item to the right of the current item satisfying all filtering conditions. In particular, if the current item is the only such item, the pointer does not move.
- Click "left". Similarly, the pointer moves to the closest item to the left of the current item satisfying all filtering conditions, and stays at the same place if the current item is the only such item.
- Add a new filtering condition (j, v), for some integers j and v. If the current item satisfies this condition, the pointer does not move. Otherwise, the pointer moves to the closest item *to the right* of the current item satisfying all filtering conditions, including the new one. If there is no such item, the operation is illegal and can not be performed.
- Remove any filtering condition (j, v) from the set. The pointer does not move.

For each ordered pair of items (i, j), answer the following question:

- If you start browsing the collection with the pointer directed at item i and with no filtering conditions in the set, what is the smallest number of operations you need to move the pointer to item j? The set of filtering conditions may be arbitrary at the end.

Input

The first line contains two integers n and m, denoting the number of items in the collection and the number of parameters each item has ($2 \le n \le 500$; $1 \le m \le 5$).

The i-th of the next n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$, denoting the parameter values of item i ($1 \le a_{i,j} \le n$).

Output

Print n lines containing n integers each.

In the i-th line, the j-th integer must be equal to the smallest number of operations required to move the pointer from item i to item j, starting with an empty set of filtering conditions.

Example

standard input	*standard output*
9 3 5 3 7 5 3 4 5 3 7 5 3 2 5 3 4 5 3 7 2 3 7 5 3 7 2 3 7	0 1 2 1 2 3 1 2 1 1 0 1 1 2 2 1 3 2 2 1 0 1 1 2 1 3 2 3 2 1 0 1 1 1 3 2 3 2 2 1 0 1 1 3 2 3 1 2 1 1 0 1 2 2 2 1 3 1 2 1 0 1 2 2 1 3 1 2 2 1 0 1 1 1 3 1 2 3 2 1 0

Note

In the example test, here is one possible fastest way to move from item 2 to item 5:

- Add a new filtering condition $(3, 4)$. Since item 2 indeed has the value of parameter 3 equal to 4, the pointer stays at item 2.
- Click "right". The pointer moves to the closest item to the right of item 2 satisfying the only active condition $(3, 4)$. This item is item 5. (Clicking "left" instead works as well.)

Here is one possible fastest way to move from item 8 to item 3:

- Add a new filtering condition $(3, 4)$. Since item 8 does not satisfy this condition, the pointer moves to the closest item to the right of item 8 with the value of parameter 3 equal to 4. This item is item 2.
- Remove the filtering condition $(3, 4)$. The pointer stays at item 2.
- Click "right". Since there are no filtering conditions in the set, the pointer moves to item 3.

Problem C. Casual Dancers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

Three friends are studying random walks. To delve deeper into the topic, they have decided to play a game.

Initially, the friends stand at integer points x_1, x_2, x_3 on the number line.

The game lasts for k seconds.

Each second, an integer j is chosen uniformly at random from the set $\{1, 2, 3\}$. Then, friend j increases their coordinate by 1 with probability p percent, or decreases their coordinate by 1 with probability $(100 - p)$ percent.

Note that multiple friends can stand at the same point, both initially and during the game.

The *stretch* is defined as the length of the shortest segment on the number line containing all three friends.

Find the expected stretch after k seconds, modulo 998 244 353 (see the Output section for details).

Input

The first line contains three integers x_1, x_2, and x_3 ($-10^5 \le x_i \le 10^5$).

The second line contains a single integer k ($1 \le k \le 2 \cdot 10^5$).

The third line contains a single integer p ($0 \le p \le 100$).

Output

Print the expected stretch after k seconds, modulo 998 244 353.

Formally, let $M = 998\,244\,353$. It can be shown that the required expected stretch can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Print the integer equal to $p \cdot q^{-1} \bmod M$. In other words, print such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Examples

standard input	*standard output*
0 0 0 1 58	1
1 2 2 1 100	332748119
5 2 3 4 50	160212060

Note

In the first example test, regardless of which friend and direction are chosen, the stretch will be equal to 1.

In the second example test, the actual expected stretch is $\frac{4}{3}$.

In the third example test, the actual expected stretch is $\frac{271}{81}$.

Problem D. Diameter Two

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are building a computer network for a new company. The network consists of n nodes numbered from 1 to n. The nodes can be connected via bidirectional wires. Each wire connects exactly two nodes. Each pair of nodes can be connected with at most one wire. If a wire connects two nodes, we'll say that these two nodes are *directly connected*.

The first k nodes (with indices $1, 2, \ldots, k$) will be *untrusted* and must be connected to the network *securely*. Each of these nodes must be directly connected to *exactly one* other node.

The remaining $n - k$ nodes (with indices $k+1, k+2, \ldots, n$) will be *trusted* and must be connected to the network *reliably*. Each of these nodes must be directly connected to *at least two* other nodes.

The *diameter* of the network must not exceed 2: for any two nodes i and j, they must either be directly connected, or there must exist a node k such that nodes i and k are directly connected, and nodes k and j are directly connected.

To minimize the costs, the number of used wires must be as small as possible.

Build a network satisfying all the conditions above, or report if this is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 50$). Description of the test cases follows.

The only line of each test case contains two integers n and k, denoting the total number of nodes and the number of untrusted nodes, respectively ($3 \le n \le 50$; $0 \le k \le n$).

Output

For each test case, if it is impossible to build a network satisfying the given conditions, print a single integer -1.

Otherwise, in the first line, print the number of used wires m. In each of the following m lines, print two integers u_i and v_i, denoting the indices of the nodes connected with the i-th wire ($1 \le u_i, v_i \le n$; $u_i \ne v_i$).

Example

standard input	*standard output*
3 3 0 5 2 6 6	3 1 2 1 3 2 3 5 1 3 2 3 3 4 3 5 4 5 -1

Problem E. Escaped from NEF

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A *cactus* is a connected undirected graph in which every edge lies on at most one simple cycle. Intuitively, a cactus is a generalization of a tree where some cycles are allowed. Multiedges (multiple edges between a pair of vertices) and loops (edges that connect a vertex to itself) are not allowed in a cactus.

You are given a *directed* graph G with n vertices with the following property. Consider an *undirected* graph G' with n vertices built as follows: for each *directed* edge (u_i, v_i) in G, add an *undirected* edge $\{u_i, v_i\}$ to G'. Then G' is a cactus.

Find the number of ordered pairs of vertices (x, y) such that there exists a path from vertex x to vertex y in G. Assume that a path from a vertex to itself always exists.

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, denoting the number of vertices and the number of edges in G ($2 \le n \le 250\,000$; $n - 1 \le m \le \left\lfloor \frac{3(n-1)}{2} \right\rfloor$).

Each of the next m lines contains two integers u_i and v_i, denoting an edge in G directed from u_i to v_i ($1 \le u_i, v_i \le n$; $u_i \ne v_i$).

The *undirected* graph consisting of *undirected* edges $\{u_i, v_i\}$ is a cactus.

It is guaranteed that the sum of n over all test cases does not exceed 250 000.

Output

For each test case, print the number of ordered pairs (x, y) such that vertex y is reachable from vertex x in G.

Example

standard input	*standard output*
2 3 3 1 2 1 3 2 3 5 5 1 2 2 3 3 4 4 5 4 2	6 18

Problem F. First Occurrence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

The famous *Thue-Morse sequence* $T = t_0 t_1 t_2 \ldots$ is an infinite binary sequence that can be defined as follows: if the number of ones in the binary representation of n is odd then $t_n = 1$, otherwise $t_n = 0$.

The sequence starts with 0110100110010110100101100110100 1 . . .

Consider a substring of this sequence $t_{l..r} = t_l t_{l+1} \ldots t_r$. Find the index of the first occurrence of $t_{l..r}$ in T. In other words, find the smallest non-negative integer i such that $t_{l..r} = t_{i..i+(r-l)}$.

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 only line of each test case contains two integers l and r ($0 \le l \le r \le 10^{18}$).

Output

For each test case, print the index of the first occurrence of $t_{l..r}$ in T.

Example

standard input	*standard output*
3 0 10 13 13 23 27	0 1 5

Note

In the first example test case, $t_{0..10}$ obviously first occurs in T at index 0.

In the second example test case, $t_{13..13} = 1$ first occurs in T at index 1.

In the third example test case, $t_{23..27} = 00110$ first occurs in T at index 5.

Problem G. Gross LCS

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	16 mebibytes (32 mebibytes for Java & Kotlin)

Note that the memory limit is unusually low.

Let $\mathrm{LCS}(A, B)$ denote the length of the longest common subsequence of integer sequences $A = \langle a_1, a_2, \ldots, a_n \rangle$ and $B = \langle b_1, b_2, \ldots, b_m \rangle$.

For an integer x, let $A + x$ denote the sequence $\langle a_1 + x, a_2 + x, \ldots, a_n + x \rangle$.

You are given two integer sequences A and B. Find the sum of $\mathrm{LCS}(A + x, B)$ over all integers x from -10^{100} to 10^{100}.

Input

The first line contains two integers n and m ($1 \le n, m \le 4000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^8 \le a_i \le 10^8$).

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($-10^8 \le b_i \le 10^8$).

Output

Print the sum of $\mathrm{LCS}(A + x, B)$ over all integers x from -10^{100} to 10^{100}.

Example

standard input	*standard output*
3 4 5 5 8 3 6 3 6	6

Note

An integer sequence P is a *subsequence* of an integer sequence Q if P can be obtained from Q by deletion of several (possibly zero or all) elements. The *longest common subsequence* of sequences A and B is the longest sequence C that is a subsequence of both A and B.

In the example test:

- $\mathrm{LCS}(A - 5, B) = \mathrm{LCS}(\langle 0, 0, 3 \rangle, \langle 3, 6, 3, 6 \rangle) = 1$;
- $\mathrm{LCS}(A - 2, B) = \mathrm{LCS}(\langle 3, 3, 6 \rangle, \langle 3, 6, 3, 6 \rangle) = 3$;
- $\mathrm{LCS}(A + 1, B) = \mathrm{LCS}(\langle 6, 6, 9 \rangle, \langle 3, 6, 3, 6 \rangle) = 2$;
- $\mathrm{LCS}(A + x, B) = 0$ for any $x \notin \{-5, -2, 1\}$.

Therefore the answer is $1 + 3 + 2 = 6$.

Problem H. Hundred Thousand Points

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	512 mebibytes

You have placed n points on a plane at coordinates $(1, 0), (2, 0), \ldots, (n, 0)$.

Informally, for each i, you draw an angle of a_i degrees from vertex $(i, 0)$ in a direction chosen uniformly at random and independently from other angles.

Formally, for each i, a **real** variable $\alpha_i \in [0; 360)$ is chosen uniformly at random, and the angle is formed by two rays drawn from the point $(i, 0)$ at polar angles of α_i and $\alpha_i + a_i$ degrees. The *interior* of the angle consists of all points located at polar angles strictly between α_i and $\alpha_i + a_i$ degrees from the point $(i, 0)$.

Two angles are considered intersecting if there exists a point belonging to the interiors of both angles.

Find the probability that no two angles intersect, modulo 998 244 353 (see the Output section for details).

Input

The first line contains a single integer n ($2 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 179$).

Output

Print the probability that no two angles intersect, modulo 998 244 353.

Formally, let $M = 998\,244\,353$. It can be shown that the required probability can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Print the integer equal to $p \cdot q^{-1} \bmod M$. In other words, print such an integer x that $0 \le x < M$ and $x \cdot q \equiv p \pmod{M}$.

Examples

standard input	*standard output*
2 90 90	686292993
3 90 90 90	982646785
3 120 30 60	795861094

Note

In the first example test, the actual probability is $\frac{5}{16}$.

In the second example test, the actual probability is $\frac{1}{64}$.

In the third example test, the actual probability is $\frac{347}{5184}$.

Problem I. Implemented Incorrectly

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider the following problem:

- *You are given a permutation $A = \langle a_1, a_2, \ldots, a_n \rangle$ containing each integer from 1 to n exactly once. Find its only cyclic shift that starts with 1.*

Consider the following algorithm to solve it:

- *Input:* $A = \langle a_1, a_2, \ldots, a_n \rangle$.
- *For each* $i = 2, 3, \ldots, n$*:*
  if $a_i < a_1$*:*
  rotate A to move a_i to the front; that is, set $A \leftarrow \langle a_i, a_{i+1}, \ldots, a_n, a_1, a_2, \ldots, a_{i-1} \rangle$.
- *Output:* $A = \langle a_1, a_2, \ldots, a_n \rangle$.

You are given a single integer n. Find the number of permutations on which the described algorithm solves the problem incorrectly.

Input

The only line contains a single integer n ($1 \le n \le 42$).

Output

Print the number of permutations on which the described algorithm works incorrectly.

Examples

standard input	*standard output*
3	1
7	1023

Note

In the first example test case, for $n = 3$, the only permutation resulting in an incorrect output is $\langle 3, 2, 1 \rangle$. The algorithm returns $\langle 2, 1, 3 \rangle$, while the correct answer is $\langle 1, 3, 2 \rangle$.

Problem J. Junk or Joy

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a positive integer k. Find the number of tuples of positive integers (n, p, m) such that $n^2 - k \cdot p^m = 1$ and p is a prime number, or report that an infinite number of such tuples exists.

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 only line of each test case contains a single integer k ($1 \le k \le 10^9$).

Output

For each test case, print the number of positive integer tuples (n, p, m) such that $n^2 - k \cdot p^m = 1$ and p is a prime, or -1 if there's an infinite number of them.

Example

standard input	*standard output*
2 5 22	3 0

Note

In the first example test case, for $k = 5$, the only possible tuples are $(4, 3, 1)$, $(6, 7, 1)$, and $(9, 2, 4)$.

In the second example test case, for $k = 22$, no possible tuples exist.

Problem K. Kilk Not

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

You are given a string s consisting of zeros (`0`), ones (`1`), and question marks (`?`).

The number of question marks in s is exactly $a + b$.

Replace a question marks with zeros and b question marks with ones to obtain a binary string t. Let $f(t)$ be the length of the longest substring of t consisting of equal digits (e.g. `11111` or `0000`).

Your task is to minimize $f(t)$.

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 three integers n, a, and b ($1 \le n \le 250\,000$; $0 \le a$; $0 \le b$).

The second line contains a string s of length n consisting of characters `0`, `1`, and `?`. The number of question marks in s is equal to $a + b$.

It is guaranteed that the sum of n over all test cases does not exceed 250 000.

Output

For each test case, print two lines.

In the first line, print a single integer $f(t)$, denoting the smallest possible length of the longest substring of t consisting of equal digits.

In the second line, print any string t achieving this value of $f(t)$ itself.

Example

standard input	*standard output*
4 7 1 2 0?01??0 10 5 0 ?000??0?0? 11 0 0 11001110100 15 2 4 ?1?11?1??11100?	1 0101010 10 0000000000 3 11001110100 4 110111101111001