

Problem A. DeepTreek

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

You are given a tree T consisting of n vertices labeled from 1 to n , rooted at 1. A pair of vertices (u, v) is called *valid* if and only if $u \neq v$, $u \neq 1$, and u is not an ancestor* of v .

For a valid pair (u, v) , define $f(u, v)$ as follows:

- Consider a new graph T' obtained by removing the edge between u and its parent, then adding a new edge between u and v . It can be proven that T' is also a tree. $f(u, v)$ is defined as the depth of T' . Here, the depth of a tree rooted at 1 is defined as $\max_{1 \leq i \leq n} d_i$, where d_i is the length of the shortest path between vertices 1 and i .

Find the sum of $f(u, v)$ over all valid pairs (u, v) .

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 an integer n ($2 \leq n \leq 2 \cdot 10^5$), representing the number of vertices in T .

The i -th of the next $n - 1$ lines contains two integers u_i and v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$), representing an edge in T . It is guaranteed that the edges form a valid tree.

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 an integer representing the sum of $f(u, v)$ over all valid pairs (u, v) .

*An ancestor of vertex v is any vertex on the simple path from v to the root, including the root, but not including v . The root has no ancestors.

Example

standard input	standard output
5	1
2	5
1 2	6
3	65
1 2	163
2 3	
3	
1 2	
1 3	
6	
1 2	
2 3	
3 4	
2 5	
5 6	
8	
1 2	
2 3	
2 4	
4 5	
5 6	
1 7	
7 8	

Problem B. Odd Cycle

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

You are given a directed graph G consisting of n vertices labeled from 1 to n . An **odd cycle** is defined as a sequence of vertices (not necessarily distinct) $p_0, p_1, \dots, p_m$, such that m is odd, and there exists a directed edge from vertex p_i to p_{i+1} for every $0 \leq i < m$.

For each vertex u , determine whether u is contained in at least one odd cycle.

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 two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 2 \cdot 10^5$), representing the number of vertices and the number of edges in G , respectively.

Each of the next m lines contains two integers u and v ($1 \leq u, v \leq n$), representing a directed edge from u to v . G **may contain** self loops and multiple edges.

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

Output

For each test case, output a line containing a binary string s of length n . For every $1 \leq i \leq n$, if vertex i is contained in at least one odd cycle, then $s_i = 1$; otherwise, $s_i = 0$.

Example

standard input	standard output
7	0
1 0	111
3 3	11
1 2	0000
2 3	11
3 1	11100010
2 2	111111000
1 1	
2 2	
4 4	
1 2	
2 3	
3 4	
4 1	
2 3	
1 2	
2 2	
2 1	
8 8	
1 2	
2 3	
3 1	
4 5	
5 4	
5 6	
6 5	
7 7	
9 14	
1 4	
4 2	
2 5	
5 3	
3 6	
6 1	
1 2	
7 8	
8 7	
8 9	
9 8	
2 4	
4 6	
6 2	

Problem C. Count Cubes

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

Xue Ba Ti! Shu Zheng Fang Ti!

There are some cubes in the 3-dimensional space. Each cube has a side length of 1. The sides of the cubes are parallel to the x , y , and z axes. A cube is said to be located at coordinates (x, y, z) if and only if its 8 corners are at:

- (x, y, z) ,
- $(x, y, z + 1)$,
- $(x, y + 1, z)$,
- $(x, y + 1, z + 1)$,
- $(x + 1, y, z)$,
- $(x + 1, y, z + 1)$,
- $(x + 1, y + 1, z)$,
- $(x + 1, y + 1, z + 1)$.

respectively.

All cubes are located at non-negative integer coordinates in space and obey the laws of gravity. Formally, if there is a cube located at (x, y, z) , then either $z = 0$ or there is a cube located at $(x, y, z - 1)$.

An x -view of a 3-dimensional structure is the 2-dimensional shape observed if you align your sight in the positive x direction. The y -view is defined similarly.

You don't know the exact locations of the cubes, but fortunately, you have the x -view and y view of the structure. You also know that the locations of all cubes satisfy: $x \leq a, y \leq b, z \leq c$. You want to find: what's the minimum and maximum number of cubes possibly present in the structure for these views to be valid? It is also possible that you have poor eyesight and there doesn't exist any valid structure, or somehow the laws of gravity are disobeyed. In any of these cases, output -1 .

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^4$). The description of the test cases follows.

The first line of each test case contains three integers a , b , and c ($1 \leq a, b, c \leq 10^3$). The size of the x -view will be $c \times b$, and the size of the y -view will be $c \times a$.

The i -th of the next c lines contains a binary string s_i of length b . If $s_j = 1$, then the x -view contains a cell at $y = j - 1$ and $z = c - i$.

The i -th of the next c lines contains a binary string s_i of length a . If $s_j = 1$, then the y -view contains a cell at $x = j - 1$ and $z = c - i$.

It is guaranteed that the sum of $(a + b) \cdot c$ over all test cases does not exceed $2 \cdot 10^6$.

Output

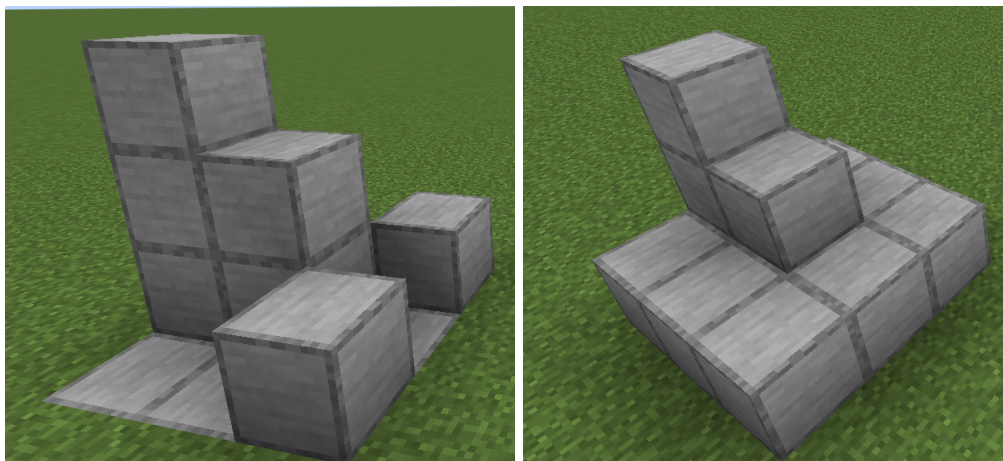
For each test case, if there doesn't exist any valid structure, output -1 . Otherwise, output two integers l and r , corresponding to the minimum and maximum number of cubes, respectively.

Example

standard input	standard output
4	7 12
3 3 3	-1
010	15 60
010	0 0
111	
100	
110	
111	
2 2 2	
11	
00	
00	
11	
5 4 3	
1111	
1111	
1111	
11111	
11111	
11111	
2 2 2	
00	
00	
00	
00	

Note

The structures of the first sample test case are given below (Source: HDZ's Minecraft):



Problem D. Xor And Mul

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

Given two non-negative integers n and m , find the number of pairs of integers (x, y) such that:

- $0 \leq x \leq n$ and $0 \leq y \leq m$.
- $(x \& y) \cdot (x \oplus y) = x \cdot y$.

where $\&$ represents the bitwise AND operation, and $\oplus$ represents the bitwise XOR operation.

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 only line of each test case contains two integers n and m ($0 \leq n, m \leq 10^9$).

Output

For each test case, output an integer representing the number of pairs of integers.

Example

standard input	standard output
5	1
0 0	3
1 1	11
3 7	166
69 96	306496
114514 191981	

Problem E. MiniC

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

In this problem, you need to write a program to solve a problem in a C-like language called MiniC.

You are given an implementation of an interpreter of MiniC, written in C++, in the contest materials section. You are also given a PDF file containing technical details about this language. You may explore freely using the provided source code.

Write a program in MiniC to solve the following problem:

- You are given a permutation $p_0, p_1, p_2, \dots, p_{n-1}$ of $\{0, 1, 2, \dots, n-1\}$. Define $f(i)$ to be the smallest j such that $j > i$ and $p_j < p_i$, or n if such j does not exist. Find $f(i)$ for all $0 \leq i < n$. The first integer of the input contains an integer n ($1 \leq n \leq 1000$). The next n integers represent $p_0, p_1, \dots, p_{n-1}$ ($0 \leq p_i < n$). You need to output n integers to standard output, where the i -th integer represents $f(i-1)$.

Your MiniC program can use at most 256 megabytes of memory and should take at most 1 second to execute. Note that if your MiniC program gets a Compile Error, Runtime Error, Time Limit Exceeded, or Memory Limit Exceeded, your submission will get the **Wrong Answer** verdict.

Output

Output the program you wrote. Your program size should not exceed 5000 characters.

Example

standard input	standard output
5	5
0 3 1 4 2	2
	5
	4
	5

Note

Note that the sample input is **what your MiniC program will receive**, and the sample output is **what your MiniC program should output**. Your submitted code, in whatever language, should print the **source code** of your MiniC program.

Problem F. Random Walk

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

There is an infinitely large 2-dimensional grid. Initially, you are at $(0, 0)$, and there is a set $S = \{(0, 0)\}$. You will walk exactly n steps on the grid. In each step:

- Suppose you are currently at (x, y) . You will walk to one of the following four positions with equal probability: $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$, $(x, y - 1)$. Let your new position be (x', y') . Then, add (x', y') to S .

Compute the expected size of S after n steps.

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

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The only line of each test case contains an integer n ($1 \leq n \leq 2 \cdot 10^5$).

Note that there is **no restriction** on the sum of n over all test cases.

Output

For each test case, output the expected size of S .

Example

standard input	standard output
4	2
1	249561091
2	499122180
3	340469626
114514	

Problem G. HDZ's Legendary Problem

Input file: `standard input`
Output file: `standard output`
Time limit: 3 seconds
Memory limit: 1024 megabytes

This is a classical game of hero versus monsters.

A monster's attributes consist of **base attributes** and **enhancement points**.

The base attributes are:

- Attack: a
- Defense: d
- Health: h
- Speed: s

The monster has n enhancement points that can be freely distributed among the four attributes a, d, h, s . Each enhancement point increases exactly one attribute by 1, and the total number of enhancement points assigned must be exactly n .

The hero has fixed attributes:

- Attack: A
- Defense: D
- Health: H
- Speed: S

The battle proceeds in rounds. In each round, both sides attack each other once **simultaneously**.

The damage dealt from one side to the other is given by

$$\text{damage} = \max \left(\left\lceil \frac{\text{own speed}}{\text{opponent speed}} \right\rceil \times \text{own attack} - \text{opponent defense}, 1 \right).$$

The battle continues until the health of at least one side becomes less than or equal to 0.

- If both sides' healths become ≤ 0 in the same round, the result is a **draw**.
- Otherwise, the side whose health becomes ≤ 0 first **loses**, and the other side **wins**.

You are given q independent queries. For each query, the hero's attributes A, D, H, S are given.

For each query, consider **all possible monsters**, i.e. all possible ways to distribute the n enhancement points among the monster's base attributes a, d, h, s . The hero will fight each of these monsters. Each of the battles is **independent**.

For each query, let the number of battles the hero wins and loses be w and l , respectively. Compute $w - l$.

Input

The first line of each test contains 5 integers a, d, h, s , and n ($1 \leq a, h, s \leq 200$, $0 \leq d, n \leq 200$), representing the base attributes of the monster and the number of enhancement points.

The following line contains an integer q ($1 \leq q \leq 10^5$), representing the number of queries.

Each of the next q lines contains 4 integers A, D, H , and S ($1 \leq A, H, S \leq 200$, $0 \leq D \leq 200$), representing the attributes of the hero.

Output

For each query, output $w - l$.

Examples

standard input	standard output
1 1 1 1 2 3 2 2 2 2 1 1 1 1 3 3 3 3	9 -4 10
10 2 9 5 10 5 1 1 1 1 10 12 8 10 19 34 78 1 10 10 10 10 100 100 100 100	-286 215 219 214 286

Note

In the first sample test, the monsters that the hero is going to fight are $(3, 1, 1, 1)$, $(1, 3, 1, 1)$, $(1, 1, 3, 1)$, $(1, 1, 1, 3)$, $(2, 2, 1, 1)$, $(2, 1, 2, 1)$, $(2, 1, 1, 2)$, $(1, 2, 2, 1)$, $(1, 2, 1, 2)$, $(1, 1, 2, 2)$.

Problem H. Distinct Substrings

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

Let Σ be an alphabet, which in this problem is the set of lowercase English letters. Let Σ^+ denote the set of all non-empty strings over Σ . For a string t , let $|t|$ denote its length, and let $t[i..j]$ denote the substring of t starting at position i and ending at position j .

You are given a string $s \in \Sigma^+$. For a string p , define $\text{occ}_s(p) = \{i \mid s[i..i + |p| - 1] = p\}$.

For all $1 \leq i \leq |s|$, compute:

$$f_s(i) = |\{w \in \Sigma^+ \mid \exists x \in \text{occ}_s(w), x \leq i \leq x + |w| - 1\}|$$

In other words, $f_s(i)$ is the number of distinct substrings of s that cover position i .

Input

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

The only line of each test case contains the string s ($1 \leq |s| \leq 10^6$).

It is guaranteed that the sum of $|s|$ over all test cases does not exceed 10^6 .

Output

For each test case, output $|s|$ integers, where the i -th integer represents $f_s(i)$.

Example

standard input	standard output
4	1
a	5 5 5 5 5
aaaaa	5 8 9 7 5
abaaa	7 12 15 15 15 12 7
abcbcba	

Problem I. Operating System

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

For an array a of length n and an integer k , define $f_k(a)$ as follows:

- Initialise a counter $c = 0$, and an empty **first-in, first-out** queue Q .
- For each $1 \leq i \leq n$:
 1. If a_i is in Q , do nothing.
 2. Otherwise, increment c by 1, and push a_i to the end of Q . If this causes the length of Q to exceed k , keep popping from the front of Q until its length is at most k .
- $f_k(a) = c$ after all elements of a are processed.

You are given two integers m and k . Construct an array a , where $1 \leq a_i \leq m$ for all $1 \leq i \leq |a|$, such that $f_k(a) < f_{k+1}(a)$.

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 only line of each test case contains two integers m and k ($1 \leq m, k \leq 10^5$).

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

Output

For each test case, if no such a exists, output -1 on a single line.

Otherwise, first output an integer n ($1 \leq n \leq 5 \cdot m$) on a single line, representing the length of a . It can be proven that if there exists a valid a , there exists one with length no larger than $5 \cdot m$.

Then, output n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq m$) on a single line.

If there are multiple valid a 's, you may output any of them.

Example

standard input	standard output
2	-1
1 1	12
5 3	4 3 2 1 4 3 5 4 3 2 1 5

Problem J. Someone's Favourite Problem

Input file: `standard input`
Output file: `standard output`
Time limit: 4 seconds
Memory limit: 256 megabytes

This is an interactive problem.

There is a hidden directed graph G consisting of n vertices. It is guaranteed that G does not contain self loops or multiple edges. Your task is to determine whether G contains a vertex with in-degree $n - 1$ and out-degree 0.

You may ask queries of the following format:

- `? u v`: does there exist an edge from vertex u to v ?

Please find any vertex that satisfies the above conditions, or determine that such a vertex does not exist, using no more than $3n$ queries.

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 an integer n ($2 \leq n \leq 10^3$).

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

Interaction Protocol

To ask a query, print a line in the following format:

- `? u v`: does there exist an edge from vertex u to v ($1 \leq u, v \leq n, u \neq v$)?

The judge will respond with an integer x . If $x = 1$, there is an edge from vertex u to v ; if $x = 0$, there is no edge from vertex u to v .

After printing each query do not forget to output the end of line and flush[†] the output. Otherwise, you will get **Idleness limit exceeded** verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive **Wrong answer** because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

To report the answer, print a line in the following format:

- `! y`: if there exists such a vertex, y ($1 \leq y \leq n$) represents its label. Otherwise, $y = -1$.

After that, proceed to process the next test case or terminate the program if it was the last test case.

Note that reporting the answer does not count towards the number of queries.

Note that the interactor is **adaptive**. This means that G may change during the interaction, but it is guaranteed that there is always some graph that satisfies the answers to all previous queries.

[†]To flush, use:

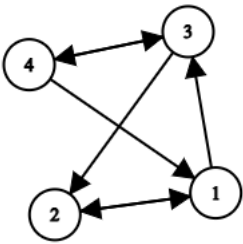
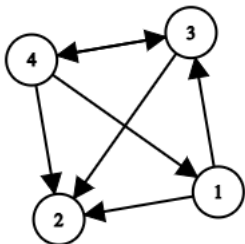
- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

Example

standard input	standard output
2	
4	
	? 1 2
1	
	? 3 1
0	
	! 2
4	
	? 1 4
0	
	? 4 3
1	
	! -1

Note

The two graphs in the sample test are given below:



Problem K. One Line

Input file: **standard input**
Output file: **standard output**
Time limit: 1.5 seconds
Memory limit: 256 megabytes

You are given an integer n . Construct a set of at least $m = \lfloor \sqrt{2n} \rfloor$ points, where the coordinates of the i -th point are (x_i, y_i) , such that:

- $x_i, y_i \in [1, n] \cap \mathbb{Z}$ for all $1 \leq i \leq m$.
- $x_i \neq x_j$ or $y_i \neq y_j$ for all $1 \leq i < j \leq m$.
- No three distinct points are collinear.

It can be shown that this is always possible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 138$). The description of the test cases follows.

The only line of each test case contains an integer n ($3 \leq n \leq 5000$). It is guaranteed that within the same test, the values of n in all test cases are **pairwise distinct**.

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

Output

For each test case, first output an integer m ($\lfloor \sqrt{2n} \rfloor \leq m \leq n^2$), representing the number of points you constructed. Then output m lines, where the i -th line contains two integers x_i and y_i ($1 \leq x_i, y_i \leq n$), representing the coordinates of the i -th point.

Example

standard input	standard output
1	4
3	1 1
	2 1
	2 2
	3 2

Problem L. Perimeter

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

There is a $n \times m$ grid of white cells. The following operation will be performed exactly k times:

- Select a cell from the $n \times m$ cells uniformly at random. If the cell is white, color it black; otherwise, do nothing.

Find the expected sum of the perimeters of all black connected components after all operations.

Formally, let $M = 10^9 + 7$. It can be shown that the exact answer can be expressed as an irreducible fraction $\frac{p}{q}$, where p and q are integers and $q \not\equiv 0 \pmod{M}$. Output the integer equal to $p \cdot q^{-1} \pmod{M}$. In other words, output such an integer x that $0 \leq x < M$ and $x \cdot q \equiv p \pmod{M}$.

Input

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

The only line in each test case contains three integers n , m , and k ($1 \leq n, m \leq 10^9, 1 \leq k \leq 10^{18}$).

Output

For each test case, output the expected sum of the perimeters of all black connected components after all operations.

Example

standard input	standard output
5	5
1 2 2	6
2 2 2	3724697
4 4 10	842233769
11 45 14	187309680
114 514 1919810	

Problem M. Classic Revisited

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

You are given an array a of length n . Find the longest even-length palindromic subsequence of a .

Input

The first line of each test contains an integer n ($2 \leq n \leq 10^5$) — the length of a .

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq 10^5$) — the elements of a .

It is guaranteed that, except for the sample test, the elements of a are generated **uniformly at random** in the range $[1, 10^5]$, and each element occurs at least **twice** in a . Specifically, a is generated by the following C++ code:

```
std::vector<int> generate(int n) {  
    std::mt19937 rng(std::chrono::steady_clock::now().time_since_epoch().count());  
    std::vector<int> a(n);  
    for (int i = 0; i < n / 2; i++) {  
        a[i] = rng() % M + 1;  
        a[i + n / 2] = a[i];  
    }  
    if (n % 2 == 1) {  
        a[n - 1] = a[rng() % (n - 1)];  
    }  
    std::shuffle(a.begin(), a.end(), rng);  
    return a;  
}
```

Output

Firstly, output an even integer m ($2 \leq m \leq n$) on a single line, representing the length of the longest even-length palindromic subsequence of a .

Then, output m integers $b_1, b_2, \dots, b_m$ ($1 \leq b_i \leq 10^5$) on a single line, representing the subsequence you found. If there are multiple longest even-length palindromic subsequences, you may output any of them.

Examples

standard input	standard output
7 1 1 4 5 1 4 5	2 5 5
9 2 10 2 10 1 9 2 1 9	4 2 10 10 2