Problem A. Classical A+B Problem

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

An integer is called a *repdigit* if it is positive and its decimal representation consists of repeated instances of the same digit. For example, 1, 666, 4444, and 999999 are repdigits, while 0, 44244, 50216, and 787788 are not.

You are given a positive integer n. It is known that n can be represented as $n = a + b$, where a and b are repdigits. Find any such representation.

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 only line of each test case contains a single integer n without leading zeros ($2 \le n < 10^{4000}$). It is guaranteed that n can be represented as $n = a + b$, where a and b are repdigits.

It is guaranteed that the total number of digits in n over all test cases does not exceed 10^5.

Output

For each test case, print two integers a and b such that $n = a + b$ and both a and b are repdigits.

If there are multiple solutions, print any of them.

Example

standard input	*standard output*
6 2 786 1332 89110 2333333 10000000000000000000000000001	1 1 777 9 333 999 88888 222 2222222 111111 99999999999999999999999999 2

Problem B. Classical Counting Problem

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

For an upcoming contest, n problems are proposed. Problem i has an initial integer score of a_i points.

There are m judges who will vote for problems they like. Each judge will choose exactly v problems, independently from other judges, and increase the score of each chosen problem by 1.

After all m judges cast their vote, the problems will be sorted in non-increasing order of score, and the first p problems will be chosen for the problemset, for some p between 1 and n. Problems with the same score can be ordered arbitrarily (this order is decided by the contest director).

How many different problemsets are possible? Print this number modulo 998 244 353. Two problemsets are considered different if some problem belongs to one of them but not to the other.

Input

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

The first line of each test case contains three integers n, m, and v, denoting the number of problems, the number of judges, and the number of problems every judge will vote for ($2 \le n \le 100$; $1 \le m \le 100$; $1 \le v \le n - 1$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the initial scores of the problems ($0 \le a_i \le 100$).

It is guaranteed that the sum of n over all test cases does not exceed 100.

Output

For each test case, print the number of possible problemsets, modulo 998 244 353.

Example

standard input	*standard output*
6 3 1 2 1 2 3 3 2 1 1 2 3 10 1 1 0 0 0 0 0 0 0 0 0 0 6 1 2 2 1 1 3 0 2 6 1 5 2 1 1 3 0 2 10 4 8 7 2 3 6 1 6 5 4 6 5	5 6 1023 23 19 240

Note

In the first test case, all possible problemsets are $\{2\}$, $\{3\}$, $\{1, 3\}$, $\{2, 3\}$, and $\{1, 2, 3\}$.

In the second test case, all possible problemsets are $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 3\}$, $\{2, 3\}$, and $\{1, 2, 3\}$.

In the third test case, any non-empty subset of problems is a possible problemset.

Problem C. Classical Data Structure Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	128 mebibytes

You have an integer array $A = [a_0, a_1, \ldots, a_{2^m-1}]$ of length 2^m. Initially, the array consists of zeros.

You also have an integer variable x. Initially, $x = 0$.

For each $i = 1, 2, \ldots, n$, you are given two integers p_i and q_i, and you have to perform the following steps:

- Let $p' = (p_i + x) \bmod 2^m$ and $q' = (q_i + x) \bmod 2^m$.
- Let $l = \min(p', q')$ and $r = \max(p', q')$.
- For each $j = l, l + 1, \ldots, r$, increase a_j by i, then increase x by a_j.

Find the value of $x \bmod 2^{30}$ at the end of this process.

Input

The first line contains two integers n and m ($1 \le n \le 500\,000$; $1 \le m \le 30$).

The i-th of the following n lines contains two integers p_i and q_i ($0 \le p_i, q_i < 2^m$).

Output

Print the value of $x \bmod 2^{30}$.

Example

standard input	*standard output*
5 2 2 1 1 3 3 2 1 0 0 2	87

Note

In the example test, initially, $A = [0, 0, 0, 0]$ and $x = 0$. Then:

- For $i = 1$, we have $l = 1$ and $r = 2$. Then, $A = [0, 1, 1, 0]$ and $x = 2$.
- For $i = 2$, we have $l = 1$ and $r = 3$. Then, $A = [0, 3, 3, 2]$ and $x = 10$.
- For $i = 3$, we have $l = 0$ and $r = 1$. Then, $A = [3, 6, 3, 2]$ and $x = 19$.
- For $i = 4$, we have $l = 0$ and $r = 3$. Then, $A = [7, 10, 7, 6]$ and $x = 49$.
- For $i = 5$, we have $l = 1$ and $r = 3$. Then, $A = [7, 15, 12, 11]$ and $x = 87$.

Problem D. Classical DP Problem

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

Let us consider a grid of squares with n rows and n columns. Arbok has cut out some part of the grid so that, for each $i = 1, 2, \ldots, n$, only the leftmost a_i squares are remaining in the i-th row from the top. The values of a_i satisfy $a_1 \le a_2 \le \ldots \le a_n$: that is, the grid looks like a Young diagram. Now, Arbok wants to place rooks into some of the remaining squares of the grid.

A rook is a chess piece that occupies one square and can move horizontally or vertically, through any number of unoccupied squares.

Let's say that a square is covered if it either contains a rook, or a rook can be moved to this square in one move.

Find r, the smallest number of rooks Arbok needs to place into some of the remaining squares so that every remaining square is covered. Also find w, the number of ways to put r rooks to satisfy the same condition, modulo 998 244 353.

Input

The first line contains a single integer n, denoting the size of the grid ($1 \le n \le 5000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the widths of the rows left by Arbok ($1 \le a_1 \le a_2 \le \ldots \le a_n \le n$).

Output

Print two integers r and w, denoting the smallest number of rooks Arbok needs to place so that every remaining square is covered, and the number of ways to put r rooks to achieve the same, modulo 998 244 353.

Example

standard input	*standard output*
3 1 2 3	2 6

Note

In the first example test, one rook is not enough to cover every square, but two rooks are enough, and there are six ways to place two rooks to cover every square (`R` denotes a rook, `*` denotes an empty square):

```
R    *    *    *    *    *
**   R*   R*   R*   *R   **
*R*  R**  *R*  **R  R**  RR*
```

Problem E. Classical FFT Problem

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

Let us consider a grid of squares with n rows and n columns. Arbok has cut out some part of the grid so that, for each $i = 1, 2, \ldots, n$, only the leftmost a_i squares are remaining in the i-th row from the top. The values of a_i satisfy $a_1 \le a_2 \le \ldots \le a_n$: that is, the grid looks like a Young diagram. Now, Arbok wants to place rooks into some of the remaining squares of the grid.

A rook is a chess piece that occupies one square and can move horizontally or vertically, through any number of unoccupied squares.

Let's say that a square is covered if it either contains a rook, or a rook can be moved to this square in one move.

Find r, the smallest number of rooks Arbok needs to place into some of the remaining squares so that every remaining square is covered. Also find w, the number of ways to put r rooks to satisfy the same condition, modulo 998 244 353.

Input

The first line contains a single integer n, denoting the size of the grid ($1 \le n < 2^{17}$).

The second line contains n integers $a_1, a_2, \ldots, a_n$, denoting the widths of the rows left by Arbok ($1 \le a_1 \le a_2 \le \ldots \le a_n \le n$).

Output

Print two integers r and w, denoting the smallest number of rooks Arbok needs to place so that every remaining square is covered, and the number of ways to put r rooks to achieve the same, modulo 998 244 353.

Example

standard input	*standard output*
3 1 2 3	2 6

Note

In the first example test, one rook is not enough to cover every square, but two rooks are enough, and there are six ways to place two rooks to cover every square (`R` denotes a rook, `*` denotes an empty square):

```
R      *      *      *      *      *
**     R*     R*     R*     *R     **
*R*    R**    *R*    **R    R**    RR*
```

Problem F. Classical Geometry Problem

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

You've just bought a new lamp that can emit light of any color! However, its controls are a bit tricky.

We will represent colors as triples (r, g, b) of real numbers between 0 and 255, inclusive. The lamp controller has eight buttons, one for each of the basic colors: black $(0, 0, 0)$, red $(255, 0, 0)$, green $(0, 255, 0)$, blue $(0, 0, 255)$, cyan $(0, 255, 255)$, magenta $(255, 0, 255)$, yellow $(255, 255, 0)$, and white $(255, 255, 255)$.

When you push the button of color c, the color of the lamp is linearly changing towards c, at the speed of 1 unit of distance per second. Once the color of the lamp reaches c, it stops changing. However, you can also stop pushing the button at any earlier moment, leaving the lamp color in some intermediate state. It is not allowed to push two or more buttons at the same time.

Formally, each button push is described by a basic color (r_c, g_c, b_c), where each of r_c, g_c, and b_c is equal to either 0 or 255, and a non-negative real number d. Let p and $c = (r_c, g_c, b_c)$ be the 3D points corresponding to the current lamp color and to the button color, and let $\overrightarrow{v} = c - p$ be the vector between them. If $p = c$, pushing this button does nothing. Otherwise, if you push the button for d seconds, the lamp color will change to $p + \frac{\overrightarrow{v}}{|\overrightarrow{v}|} \cdot \min(d, |\overrightarrow{v}|)$.

You are given a target color (r, g, b), where r, g, and b are integers. Initially, the lamp is black. Find any sequence of at most 10 button pushes that changes its color to (r, g, b). It is guaranteed that such a sequence always exists.

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 only line of each test case contains three integers r, g, and b, denoting the target color ($0 \le r, g, b \le 255$).

Output

For each test case, print an integer m, denoting the number of button pushes in your sequence ($0 \le m \le 10$).

Then, print m lines, describing button pushes in your sequence in chronological order. Each of these lines must contain three integers r_c, g_c, and b_c, followed by a real number d ($r_c, g_c, b_c \in \{0, 255\}$; $0 \le d \le 10^4$).

Your answer will be considered correct if the distance between point (r, g, b) and the point corresponding to the lamp color after your sequence does not exceed 10^{-6}.

You do not have to find the shortest sequence. If there are multiple solutions, print any of them.

Example

standard input	*standard output*
3 105 255 175 174 174 174 0 0 0	3 0 255 0 3000.0 0 255 255 119.0 255 255 255 119.0 1 255 255 255 301.376840517 2 255 255 255 50.216 0 0 0 58.0

Problem G. Classical Graph Theory Problem

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

Let $G = (V, E)$ be a connected undirected graph.

A set of vertices S is called a *dominating set* if every vertex $v \in V$ either belongs to S, or has a neighbor in S.

A vertex v is called a *leaf* if it has exactly one neighbor.

Graph G satisfies the following property: every vertex has at most two neighboring leaves.

Find a set $S \subset V$ such that:

- S is a dominating set in G;
- $V \setminus S$ is a dominating set in G;
- $|S| = \lfloor \frac{|V|}{2} \rfloor$.

It is guaranteed that such a set always exists.

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, denoting the number of vertices and the number of edges in G ($2 \le n \le 2 \cdot 10^5$; $1 \le m \le 5 \cdot 10^5$).

Each of the next m lines contains two integers x_i and y_i, denoting the endpoints of the i-th edge ($1 \le x_i, y_i \le n$; $x_i \ne y_i$). The graph does not contain loops or multiple edges. Every vertex has at most two neighboring leaves.

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

Output

Print $\lfloor \frac{n}{2} \rfloor$ distinct integers $s_1, s_2, \ldots, s_{\lfloor n/2 \rfloor}$, denoting the vertices belonging to S in any order ($1 \le s_i \le n$).

If there are multiple solutions, print any of them.

Example

standard input	*standard output*
2 6 7 1 2 1 3 2 3 3 4 4 5 4 6 5 6 3 2 1 2 2 3	2 3 6 2

Problem H. Classical Maximization Problem

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

You are given $2n$ distinct points on a plane. Point i has integer coordinates (x_i, y_i).

Points i and j are a *friendly pair* if either $x_i = x_j$ or $y_i = y_j$.

Form n pairs of points. Every point must belong to exactly one pair. The number of friendly pairs among your n pairs must be maximized.

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 i-th of the next $2n$ lines contains two integers x_i and y_i, denoting the coordinates of the i-th point ($-10^9 \le x_i, y_i \le 10^9$). All points are distinct.

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

Output

For each test case, print a non-negative integer k, denoting the maximum possible number of friendly pairs.

In the i-th of the next n lines, print two integers a_i and b_i, denoting a pair formed by points a_i and b_i ($1 \le a_i, b_i \le 2n$; $a_i \ne b_i$).

Every integer from 1 to $2n$ must appear among a_i and b_i exactly once. The number of indices i such that points a_i and b_i are a friendly pair must be equal to k.

Example

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

Problem I. Classical Minimization Problem

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

You are given $2n$ distinct points on a plane. Point i has integer coordinates (x_i, y_i).

Points i and j are a *friendly pair* if either $x_i = x_j$ or $y_i = y_j$.

Form n pairs of points. Every point must belong to exactly one pair. The number of friendly pairs among your n pairs must be minimized.

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 i-th of the next $2n$ lines contains two integers x_i and y_i, denoting the coordinates of the i-th point ($-10^9 \le x_i, y_i \le 10^9$). All points are distinct.

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

Output

For each test case, print a non-negative integer k, denoting the minimum possible number of friendly pairs.

In the i-th of the next n lines, print two integers a_i and b_i, denoting a pair formed by points a_i and b_i ($1 \le a_i, b_i \le 2n$; $a_i \ne b_i$).

Every integer from 1 to $2n$ must appear among a_i and b_i exactly once. The number of indices i such that points a_i and b_i are a friendly pair must be equal to k.

Example

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

Problem J. Classical Scheduling Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	512 mebibytes

You have t minutes until a very important exam in a very important subject. You have a list of n different topics of the subject, numbered from 1 to n. It takes a_i minutes to learn the i-th topic.

However, the subject is pretty complicated, and even if you learn some topic, you might not necessarily be confident about it during the exam. You know that to be confident about the i-th topic, you need to learn at least b_i topics in total (including topic i).

It takes no time to switch from one topic to another while learning. Thus, you have enough time to learn distinct topics $p_1, p_2, \ldots, p_k$ if $a_{p_1} + a_{p_2} + \ldots + a_{p_k} \le t$, and you will be confident about topic p_i during the exam if $k \ge b_{p_i}$. Note that the order in which you learn the topics is not important.

You want to figure out what topics you should learn before the exam to maximize the number of topics you will be confident about.

Input

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

The first line of each test case contains two integers n and t, denoting the number of topics and the amount of time you have until the exam ($1 \le n \le 2 \cdot 10^5$; $1 \le t \le 2 \cdot 10^{14}$).

The i-th of the next n lines contains two integers a_i and b_i, denoting the amount of time it takes to learn the i-th topic and the total number of topics you need to learn to be confident about the i-th topic ($1 \le a_i \le 10^9$; $1 \le b_i \le n$).

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

Output

For each test case, print the maximum number of topics you can be confident about during the exam.

Then print k, denoting the number of topics you should learn, followed by k distinct integers $p_1, p_2, \ldots, p_k$ in any order, denoting the indices of the topics you should learn ($0 \le k \le n$; $1 \le p_i \le n$).

If there are multiple solutions, print any of them.

Example

standard input	*standard output*
2 4 100 20 1 40 4 60 3 30 3 1 5 10 1	2 3 1 2 4 0 0

Note

In the first test case, you should learn topics 1, 2, and 4. This will take you $20 + 40 + 30 = 90$ minutes, which is fine since you have 100 minutes until the exam. Even though you will not be confident about topic 2 (you would need to learn all four topics for that), you will be confident about topics 1 and 4. It is impossible to be confident about more than two topics in this test case.

Problem K. Classical Summation Problem

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

There are n cities and $n-1$ roads in the Republic of Never. The cities are conveniently numbered from 1 to n. The roads are numbered from 1 to $n-1$, and road i connects cities i and $i+1$. Every road can be traversed in both directions. The distance $d(u,v) = |u-v|$ is defined as the smallest number of roads one needs to use to move from city u to city v.

There are k friends who are looking to meet. The i-th friend lives in city a_i. For the meeting, the friends will choose city v such that $\sum_{i=1}^{k} d(v, a_i)$ is minimum. If there are several such cities, they will choose the one with the smallest number among them.

Unfortunately, you know just the number of friends but nothing about the cities where they live. Every friend might live in any of the n cities; hence, there are n^k options overall. You would like to find the sum of the numbers of cities the friends will choose in all the n^k options. Output this sum modulo 998 244 353.

Input

The only line contains two integers n and k, denoting the number of cities and the number of friends ($2 \le n, k \le 10^6$).

Output

Print the sum of the numbers of cities the friends will choose in all the n^k options, modulo 998 244 353.

Examples

standard input	*standard output*
3 2	14
5 3	375

Note

In the first example, with three cities and two friends, there are $3^2 = 9$ options to consider. If either friend lives in city 1, the friends will choose city 1, and there are 5 such options. Otherwise, if either friend lives in city 2, the friends will choose city 2, and there are 3 such options. In the remaining case, if both friends live in city 3, they will choose city 3, and there is 1 such option. The total is $5 \cdot 1 + 3 \cdot 2 + 1 \cdot 3 = 14$.