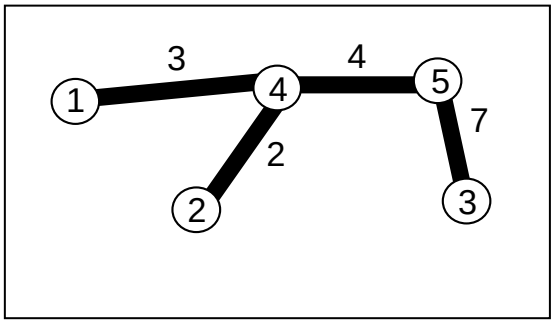
2nd INTERNATIONAL TOURNAMENT IN INFORMATICS "JOHN ATANASOV"
Shumen, 27.11.2010 г.

Group B - Juniors

Task B1. RECONSTRUCTION

The young programmers Peter and Stancho were hired by two cosmic agencies. The agency of Peter designed a new cosmic station, composed of *N* modules, labeled from 1 to *N*. Some couples of different modules are linked by corridors in such way that it is possible to go from each module to each other module by unique path of corridors (see the Figure). The length of each of the corridors is positive integer. There is no more then one corridor that links two modules. The chiefs of Peter would like to keep in secret the project. That is why Peter encoded the topology of the station giving for each two modules the distance between them (i.e. the sum of the lengths of the corridors on the unique path between the modules).



Now Stancho has a difficult task – he promised to his bosses to decrypt the coding of Peter and to reconstruct the topology of the station. Unfortunately, Stancho is not experienced enough. Help him. Write a program **recon** that solves the task.

Input

The first line of the standard input contains the number *N* of the modules ($3 \le N \le 1024$). Then $N - 1$ lines follow. On the first of these lines the distances from module 1 to modules 2, 3,…, *N* are given, separated by single spaces. On the second line re given, separated by single spaces also, the distances from module 2 to modules 3, 4,…, *N*, and so on. The last line contains only the distance from module $N - 1$ to module *N*. All distances are positive integers not grater than 1024.

Output

The program has to print *N* lines to the standard output. The first line has to contain the list of the neighbors of module 1, i.e. the modules that are linked with corridors to it. The list has to start with the number *L* of neighbors followed by their labels, sorted in increasing order. All numbers has to be separated by single spaces. On the second row of the output, formatted in the same way, the list of the neighbors of module 2 has to be printed, and so on. The output has to finish with the list of the neighbors of module *N*.

EXAMPLE

Input	Output
5 5 14 3 7 13 2 6 11 7 4	1 4 1 4 1 5 3 1 2 5 2 3 4

2nd INTERNATIONAL TOURNAMENT IN INFORMATICS "JOHN ATANASOV"
Shumen, 27.11.2010 г.

Group B - Juniors

Task B2. COMPETITION

K competitors are taking part in a competition. Each of them must finish N full laps of a round circuit. All the competitors start together from the starting line. At the start of the event every runner is in his 'normal' form. While making his laps he is losing his stamina and he is running with lower and lower tempo. That lower tempo means that every lap is done 1 millisecond slower than the previous one. When in normal form competitor with number *i* is doing one lap for ms_i milliseconds (ms_i is a positive integer). The rules of the competition allow a positive integer p_i ($1 \le p_i \le N$) to be announced for every competitor before the start. On every full p_i laps done the runner gets an energy drink (while passing the starting line) which returns him to full strength (after that his stamina is dropping again the same way as before). The drinking takes 0 time. It's obvious that while doing N laps every competitor will cross the starting line N times (we count the crossing after the last lap, but do not count the crossing of the line when they start the run).

Write program **competition,** which determines the maximum number of competitors who will cross the starting line together at any point of the competition (***as we are working in milliseconds "together" means after equal amount of milliseconds after the competition's start***)

Input

On the first line of the standard input you must read two positive integers, separated with an interval: *K* – competitors count and *N* – laps count.

After that there are K rows (one for each runner). On every row there are two positive integers: ms_i - milliseconds for competitor ***i*** to run one lap in "normal" form and p_i – lap count on which the runner receives his energy drinks and returns to "normal" form.

Output

Print one single integer – the determined maximum count of competitors who will cross the starting line together at some moment of the competition..

Restrictions

$2 \le K \le 10\,000$; $1 \le N \le 1\,000$; $1 \le ms_i \le 1\,000\,000$; $1 \le p_i \le N$; Memory: 1 MB

EXAMPLE

Input	Output
4 3 26 2 39 3 45 1 56 2	2

Example's explanation: The runners will do the following laps:
Runner1 – for 26, 27 and 26 msec.; Runner2 – for 39, 40 and 41 msec.; Runner3– for 45, 45 and 45 msec.; Runner4 – for 56, 57 and 56 msec. Respectively they will cross the starting line at: Runner1 – after 26, 53, 79 msec.; Runner2– after 39, 79 and 120 msec; Runner3 – after 45, 90 and 135 msec; Runner4 – after 56, 113 and 169 msec. The only case when more than one runner will cross the starting line is after 79 milliseconds when runner1 and runner2 will be on the line together.

Grading: 20% of the test cases will have **$1 \le ms_i \le 100$.**

2nd INTERNATIONAL TOURNAMENT IN INFORMATICS "JOHN ATANASOV"
Shumen, 27.11.2010 г.

Group B - Juniors

Task B3. EXAMS

Ivan has finally been accepted in university! After an exhausting year of preparation for the entry exams, he was determined to take a big break. Unfortunately the summer vacation was short and wasn't enough for him to take a proper rest, but he didn't get disappointed. Ivan had heard that when you are in university it isn't compulsory to attend to every single lecture. He decided to lengthen his vacation, even though the academic year had started. And so did he!

But, of course, for every plus there is a minus – as well as in secondary schools in every college or university there are exams. More precisely in Ivan's curriculum there are exactly **N** exams. They may vary in difficulty, and therefore two exams may give different amount of points. Ivan is not obliged to take all exams, but the points gathered from the tests are important to him, because they are part of his final grade. He quickly realized that he needs at least **T** points from all exams, so that eventually he has a good grade. He is sure that he can make at least **T** points after attending to all the exams, but he suspects that there might be different ways to do this. The more ways there are, the more free time he has - Ivan will be able to choose which exam to take and which not to. In this manner he will decide how to distribute his leisure time – for instance which concerts or football matches to visit. Actually Ivan is completely sure in his abilities, so if he takes an exam, he will get the full score. Don't let Ivan wonder in how many ways he can manage to gather at least **T** points for too long, but write a program **`points`** that finds out this number for him.

Input
The first row of the standard input will contain two positive integers **N (N ≤ 36)** and **T**. On the next row there are going to be **N** positive integers, split by a single space, which represent the points of each exam. Every number in the second row will not be greater than $\mathbf{10^{13}}$.

Output
On the standard output print a single integer – the number of ways in which Ivan can choose which exams to take and which not, so that the sum of the gathered points to be at least **T**.

EXAMPLE 1

Input	Output
4 6 1 2 5 4	9

Explanation: there are nine ways in which Ivan can choose which exam to take and which not to, so that the sum of the gathered points to be at least 6: (first and second); (second and third); (first, second u third); (second u fourth); (first, second u fourth); (first, third u fourth); (second, third u fourth); (first, second, third u fourth); (third u fourth).

EXAMPLE 2

Input	Output
8 90 1000 2 5 79 12 3 1 3	166

**In* 20*% of the tests the numbers in the second row of the input will be* ≤ 1000.