Task Overview Sheet

	Archery	Hiring	POI	Raisins
Type	Batch	Batch	Batch	Batch
Detailed Feedback	Partial	None	**Full**	Partial
Time Limit (per test case)	2 seconds	1.5 seconds	2 seconds	5 seconds
Memory Limit (per test case)	64 MB	64 MB	64 MB	128 MB
Points	100	100	100	100
	400			

C++ programmers should be aware that using C++ streams (cin / cout) may lead to I/O bottlenecks and substantially slower performance. Please see the technical info sheet for ways to avoid this.

TECHNICAL INFO SHEET

These pages contain helpful information on how to avoid slow input/output performance with C++ streams (cin / cout), how to use 64-bit data types (variables) and how to flush the output for interactive tasks. They also include reference for what options are given to the compilers and what stack limitations are in place.

Slow Input / Output with C++ Streams

When solving tasks with very large amounts of input / output data, you may notice that C++ programs using the cin and cout streams are much slower than equivalent programs that use the scanf and printf functions for input and output processing. Thus, if you are using the cin / cout streams we strongly recommend that you switch to using scanf / printf instead. However, if you still want to use cin / cout, we recommend adding the following line at the beginning of your program:

```
ios::sync_with_stdio(false);
```

and also making sure that you never use `endl`, but use `"\n"` instead.

Please note, however, that including `ios::sync_with_stdio(false)` breaks the synchrony between cin / cout and scanf / printf, so if you are using this, you should never mix usage of cin and scanf, nor mix cout and printf.

64-bit Data Types

For some tasks you may need to deal with numbers too large to fit in 32 bits. In these cases, you would have to use a 64-bit integer data type, such as `long long` in C/C++ or `int64` in Pascal. Here is some sample code that illustrates the usage of these data types:

C/C++

```
int main(void) {
    long long varname;
    scanf("%lld", &varname);
    // Do something with the varname variable
    printf("%lld\n", varname);
    return 0;
}
```

Pascal

```
var
    varname: Int64;
begin
    read(varname);
```

```
    { Do something with the varname variable }
    writeln(varname);
end.
```

Flushing the Output

Whenever you solve an interactive task, you always need to flush the buffer of your output after every new line printed on the output. Here is some code to illustrate how to do this under C, C++ and Pascal:

C or C++ with scanf / printf

```
fflush(stdout);
```

C++ with cin / cout

```
cout << flush;
```

Pascal

```
flush(output);
```

Compiler Options

The following commands will be used to compile solutions of batch and interactive tasks (say the task name is abc):

C

```
gcc –o abc abc.c –std=gnu99 –O2 –s –static –lm –x c
```

C++

```
g++ –o abc abc.cpp –O2 –s –static –lm –x c++
```

Pascal

```
fpc –O2 –XS –Sg abc.pas
```

Stack Limitations

Whenever your program is executed through the contest system, the stack size will only be limited by the memory limit for the corresponding task.

ARCHERY

An archery tournament is held according to the following rules. There are ***N*** targets arranged in a line and numbered from 1 to ***N*** inclusive according to their place on the line (the leftmost target being target 1, and the rightmost target being target ***N***). There are also ***2*N*** archers. At any time during the tournament, there are two archers on each target. Every round of the tournament goes according to the following procedure:

- The two archers on each target compete with each other and determine a winner and a loser between them. Then all archers are rearranged as follows:
  - The winners on targets 2 to ***N*** inclusive move to the target on their left (i.e., targets 1 to ***N - 1*** respectively).
  - The losers on targets 2 to ***N*** inclusive, as well as the winner on target 1, remain on the same target.
  - The loser on target 1 moves to target ***N***.

The tournament continues for ***R*** rounds, with the number of rounds being at least as many as the number of archers (i.e., ***R*** ≥ ***2*N***).

You are the only archer to arrive for the tournament exactly on time. All other ***2*N - 1*** archers have arrived early and are already standing in a line. What you have to do now is to insert yourself somewhere into the line amongst them. You know that after you take your position, the two leftmost archers in the line will start the tournament on target 1, the next two will start on target 2 and so on, with the two rightmost archers starting on target ***N***.

All the ***2*N*** archers in the tournament (including yourself) are ranked by skill, where a smaller rank corresponds to better skill. No two archers have the same rank. Also, whenever two archers compete, the one with the smaller rank will always win.

Knowing how skilled each of your competitors is, you want to insert yourself in such a way as to ensure that you will finish the tournament on a target with as small a number as possible. If there are multiple ways to do this, you prefer the one that starts at a target with as large a number as possible.

TASK
Write a program that, given the ranks of all archers, including yourself, as well as your competitors' arrangement on the line, determines on which target you should start the tournament, so that you can achieve your goals as defined above.

CONSTRAINTS

$1 \le N \le 200{,}000$ The number of targets; also equal to half the number of archers

$2*N \le R \le 1{,}000{,}000{,}000$ The number of tournament rounds

$1 \le S_k \le 2*N$ The rank of archer k

INPUT

Your program must read from standard input the following data:

- The first line contains the integers N and R, separated by a space.
- The next $2*N$ lines list the ranks of the archers. The first of these lines contains your rank. The rest of these lines contain the ranks of the other archers, one archer per line, in the order in which they have arranged themselves (from left to right). Each of these $2*N$ lines contains a single integer between 1 and $2*N$ inclusive. A rank of 1 is the best and a rank of $2*N$ is the worst. No two archers have the same rank.

OUTPUT

Your program must write to standard output a single line containing a single integer between 1 and N inclusive: the number of the target on which you will start the tournament.

GRADING

For a number of tests, worth a total of 60 points, N will not exceed 5,000.
Also, for some of these tests, worth a total of 20 points, N will not exceed 200.

EXAMPLES

Sample Input	Sample Output
4 8 7 4 2 6 5 8 1 3	3

You are the second worst archer. If you start on target 1, you will then go to target 4 and stay there until the end. If you start on target 2 or 4, you will just stay there for the whole tournament. If you start on target 3, you will beat the worst archer and then move to target 2 and stay there.

Sample Input	Sample Output
4 9 2 1 5 8	2

3 4 7 6	

You are the second best archer. The best one is already on target 1 and will stay there for the whole duration of the tournament. Thus, no matter where you start, you will always move from your target, going through all targets from 4 to 1 over and over again. In order for you to end on target 1 after 9 transitions, you have to start on target 2.

HIRING

You have to hire workers for a construction project. There are N candidates applying for the job, numbered from 1 to N inclusive. Each candidate k requires that if he is hired, he must be paid at least S_k dollars. Also, each candidate k has a qualification level Q_k. The regulations of the construction industry require that you pay your workers in proportion to their qualification level, relative to each other. For example, if you hire two workers A and B, and $Q_A = 3 * Q_B$, then you have to pay worker A exactly three times as much as you pay worker B. You are allowed to pay your workers non-integer amounts of money. This even includes quantities that cannot be written with a finite number of digits in decimal form, such as a third or a sixth of a dollar.

You have W dollars at hand and you want to hire as many workers as possible. You decide whom to hire and how much to pay them, but you have to meet the minimum salary requirements of those you choose to hire, and you have to obey the industry regulations. You also have to fit within your budget of W dollars.

The nature of your project is such that the qualification level is completely irrelevant, so you are only interested in maximizing the number of workers without regard to their qualification level. However, if there is more than one way to achieve this, then you want to select the one where the total amount of money you have to pay your workers is as small as possible. In case there is more than one way to achieve this, then you are indifferent among these ways and you would be satisfied with any one of them.

TASK

Write a program that, given the different salary requirements and qualification levels of the candidates, as well as the amount of money you have, determines which candidates you should hire. You must hire as many of them as possible and you must do so with as little money as possible, while complying with the industry regulations specified above.

CONSTRAINTS

$1 \le N \le 500{,}000$	The number of candidates
$1 \le S_k \le 20{,}000$	The minimum salary requirement of candidate k
$1 \le Q_k \le 20{,}000$	The qualification level of candidate k
$1 \le W \le 10{,}000{,}000{,}000$	The amount of money available to you

IMPORTANT NOTE

The maximum value of W does not fit in 32 bits. You have to use a 64-bit data type, such as `long long` in C/C++ or `int64` in Pascal, in order to store the value of W in a single variable. Please see the technical info sheet for details.

INPUT

Your program must read from standard input the following data:

•The first line contains the integers ***N*** and ***W***, separated by a space.
•The next ***N*** lines describe the candidates, one candidate per line. The k^{th} of these lines describes candidate number ***k*** and it contains the integers S_k and Q_k, separated by a space.

OUTPUT
Your program must write to standard output the following data:
•The first line must contain a single integer ***H***, the number of workers that you hire.
•The next ***H*** lines must list the identifying numbers of the candidates you choose to hire (each of them a different number between 1 and ***N***), one per line, in any order.

GRADING
For any given test case, you will receive full points if your choice of candidates enables you to achieve all of your goals, while satisfying all constraints. If you produce an output file with a correct first line (i.e., a correct value of ***H***), but which does not meet the above description, you will receive 50% of the points for that test case. The latter will be the case even if the output file is not properly formatted, as long as the first line is correct.

For a number of tests, worth a total of 50 points, ***N*** will not exceed 5,000.

EXAMPLES

Sample Input	Sample Output
4 100 5 1000 10 100 8 10 20 1	2 2 3

The only combination for which you can afford to hire two workers and still meet all the constraints is if you select workers 2 and 3. You can pay them 80 and 8 dollars respectively and thus fit in your budget of 100.

Sample Input	Sample Output
3 4 1 2 1 3 1 3	3 1 2 3

Here you can afford to hire all three workers. You pay 1 dollar to worker 1 and 1.50 dollars each to workers 2 and 3, and you manage to hire everyone with the 4 dollars that you have.

Sample Input	Sample Output
3 40 10 1 10 2	2 2 3

10 3	

Here you cannot afford to hire all three workers, as it would cost you 60 dollars, but you can afford to hire any two of them. You choose to hire workers 2 and 3 because they would cost you the smallest sum of money, compared to the other two-worker combinations. You can pay 10 dollars to worker 2 and 15 dollars to worker 3 for a total of 25 dollars. If you were to hire workers 1 and 2 you would have to pay them at least 10 and 20 dollars respectively. If you were to hire 1 and 3, then you would have to pay them at least 10 and 30 dollars respectively.

POI

Full Feedback Problem

The local Plovdiv Olympiad in Informatics (POI) was held according to the following unusual rules. There were ***N*** contestants and ***T*** tasks. Each task was graded with only one test case, therefore for every task and every contestant there were only two possibilities: either the contestant solved the task, or the contestant did not solve the task. There was no partial scoring on any task.

The number of points assigned to each task was determined after the contest and was equal to the number of contestants that did not solve the task. The score of each contestant was equal to the sum of points assigned to the tasks solved by that contestant.

Philip participated in the contest, but he is confused by the complicated scoring rules, and now he is staring at the results, unable to determine his place in the final standings. Help Philip by writing a program that calculates his score and his ranking.

Before the contest, the contestants were assigned unique IDs from 1 to ***N*** inclusive. Philip's ID was ***P***. The final standings of the competition list the contestants in descending order of their scores. In case of a tie, among the tied contestants, those who have solved more tasks will be listed ahead of those who have solved fewer tasks. In case of a tie by this criterion as well, the contestants with equal results will be listed in ascending order of their IDs.

TASK

Write a program that, given which problems were solved by which contestant, determines Philip's score and his rank in the final standings.

CONSTRAINTS

$1 \le N \le 2{,}000$ The number of contestants
$1 \le T \le 2{,}000$ The number of tasks
$1 \le P \le N$ Philip's ID

INPUT

Your program must read from standard input the following data:

- The first line contains the integers ***N***, ***T*** and ***P***, separated by individual spaces.
- The next ***N*** lines describe which tasks were solved by which contestant. The k^{th} of these lines describes which tasks were solved by the contestant with ID ***k***. Each such line contains ***T*** integers, separated by spaces. The first of these numbers denotes whether or not contestant ***k*** solved the first task. The second number denotes the same for the second task and so on. These ***T*** numbers are all either 0 or 1, where 1 means that contestant ***k*** solved the corresponding task, and 0 means that he or she did not solve it.

OUTPUT

Your program must write to standard output a single line with two integers separated by a single space. First, the score that Philip got on the POI competition. Second, Philip's rank in the final standings. The rank is an integer between 1 and ***N*** inclusive, with 1 denoting the contestant listed at the top (i.e., a contestant who has the highest score) and ***N*** to the one listed at the bottom (i.e., a contestant with the lowest score).

GRADING

For a number of tests, worth a total of 35 points, no other contestant will have the same score as Philip.

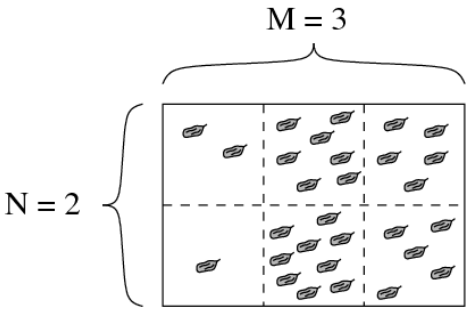
EXAMPLE

Sample Input	Sample Output
5 3 2 0 0 1 1 1 0 1 0 0 1 1 0 1 1 0	3 2

The first problem was unsolved by only one contestant, so it is worth 1 point. The second problem was unsolved by two contestants, so it is worth 2 points. The third problem was unsolved by four contestants, so it is worth 4 points. Thus the first contestant has a score of 4; the second contestant (Philip), the fourth and the fifth contestants all have a score of 3; and the third contestant has a score of 1. Contestants 2, 4 and 5 are all tied according to the first tie-break rule (number of problems solved), and according to the second tie-break rule (smaller ID) Philip ranks before the others. Thus Philip's rank in the final standings is 2. He is only behind the contestant with ID 1.

RAISINS

Plovdiv's famous master chocolatier Bonny needs to cut a slab of chocolate with raisins. The chocolate is a rectangular block of identical square pieces. The pieces are aligned with the edges of the chocolate, and they are arranged in ***N*** rows and ***M*** columns, for a total of ***N*M*** pieces. Each piece has one or more raisins on it, and no raisins lie between or across pieces.



Initially, the chocolate is one single, monolithic block. Bonny needs to cut it into smaller and smaller blocks until finally she has cut the chocolate down to its ***N*M*** individual pieces. As Bonny is very busy, she needs the help of her assistant, Sly Peter, to do the cutting. Peter only makes straight line, end-to-end cuts and he wants to be paid for every single cut he makes. Bonny has no money at hand, but she has plenty of raisins left over, so she offers to pay Peter in raisins. Sly Peter agrees to this arrangement, but under the following condition: every time he cuts a given block of chocolate into two smaller blocks, he has to be paid as many raisins as there are on the block he was given.

Bonny wants to pay Peter as little as possible. She knows how many raisins there are on each of the ***N*M*** pieces. She can choose the order in which she gives Peter any remaining blocks, and she can also tell Peter what cuts to make (horizontal or vertical) and where exactly to make them. Help Bonny decide how to cut the chocolate into individual pieces, so that she pays Sly Peter as few raisins as possible.

TASK

Write a program that, given the number of raisins on each of the individual pieces, determines the minimum number of raisins that Bonny will have to pay Sly Peter.

CONSTRAINTS

$1 \le N, M \le 50$ The number of pieces on each side of the chocolate

$1 \le R_{k,p} \le 1000$ The number of raisins on the piece in the k^{th} row and the p^{th} column

INPUT

Your program must read from standard input the following data:

- The first line contains the integers ***N*** and ***M***, separated by a single space.
- The next ***N*** lines describe how many raisins there are on each piece of the chocolate. The k^{th} of these ***N*** lines describes the k^{th} row of the chocolate. Each such line contains ***M*** integers separated by single spaces. The integers describe the pieces on the corresponding row in order from left to right. The p^{th} integer on the k^{th} line (among these ***N*** lines) tells you how many raisins are on the piece in the k^{th} row and the p^{th} column.

OUTPUT

Your program must write to standard output a single line containing a single integer: the minimum possible number of raisins that Bonny would have to pay Sly Peter.

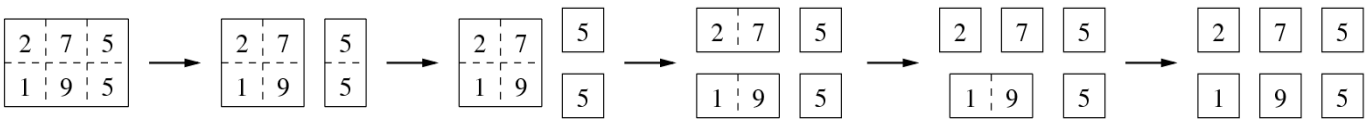
GRADING

For a number of tests, worth a total of 25 points, ***N*** and ***M*** will not exceed 7.

EXAMPLE

Sample Input	Sample Output
2 3 2 7 5 1 9 5	77

One possible way (out of many) to achieve a cost of 77 is as follows:



The first cut that Bonny asks Peter to make separates the third column from the rest of the chocolate. Bonny needs to pay Peter 29 raisins for this.

Then Bonny gives Peter the smaller of the two blocks: the one that has two pieces with 5 raisins each, and asks Peter to cut the block in two in exchange for 10 raisins.

After this, Bonny gives Peter the largest remaining block: the one having pieces with 2, 7, 1 and 9 raisins respectively. Bonny asks Peter to cut it horizontally, separating the first and the second row and pays him 19 raisins.

Following this, Bonny gives Peter the top-left block, paying 9 raisins. Finally, Bonny asks Peter to split the bottom-left block, paying 10 raisins.

The total cost to Bonny is 29 + 10 + 19 + 9 + 10 = 77 raisins. No other cutting arrangement can get the chocolate cut into its 6 pieces at a smaller cost.