Indian National Olympiad in Informatics, 2005

Time: 3 hours *27 April, 2005*

Instructions

(a) You will have to return this question paper at the end of the examination with relevant parts filled out.

(b) There are two questions. You have to write working programs in Pascal, C or C++ to solve each of these questions. Only your source code will be submitted for evaluation. Your program will be recompiled and run on the evaluation computer.

Make sure your C/C++ programs compile with the GNU C compiler (`djgpp`, under Windows). Programs written using Turbo C++ may not compile and run in the evaluation environment and may hence lose marks.

At the end of each question, there is a space to indicate the location of the source code file for your solution. Please fill up this information without fail. Otherwise, your solution cannot be evaluated.

(c) All input for your programs will come from the keyboard. All output from your programs should be written to the screen.

(d) Please fill out your contact details below as completely as you can. You have been assigned a fresh roll number for INOI. Ask your centre supervisor if you do not know your roll number.

Contact details	
Roll Number:	
Name:	
Address:	PIN Code
Phone Number(s): (with STD code)	
Email address:	

Question 1 Frog jumping

The latest hit on TV is a jumping game played on a giant rectangular chessboard. Each participant dresses up in a green frog suit and starts at the top left corner of the board. On every square there is a spring-loaded launcher that can propel the person either to the right or down.

Each launcher has two quantities R and D associated with it. The launcher can propel the person upto R squares to the right and upto D squares down. The participant can set the direction of the launcher to *Right* or *Down* and set the number of squares to jump to any number between 1 and R squares when jumping right, or between 1 and D squares when jumping down. The winner is the one who can reach bottom right corner of the chessboard in the smallest number of jumps.

For instance, suppose you have 3×4 chessboard as follows. In each square, the pair of numbers indicates the quantities (R, D) for the launcher on that square.

$(1,2)$	$(1,2)$	$(1,2)$	$(2,1)$
$(3,1)$	$(1,1)$	$(1,2)$	$(1,2)$
$(1,1)$	$(1,1)$	$(1,2)$	$(2,2)$

Here, one way to reach the bottom right corner is to first jump 1 square right, then jump 2 squares down to the bottom row, then jump right two times, one square a time, for a total of 4 jumps. Another way is to first jump 1 square down, then jump 3 squares right to the last column and finally jump one square down to the bottom right corner, for a total of 3 jumps. On this board, it is not possible to reach the bottom right corner in fewer than 3 jumps.

Your task is to write a program to calculate the smallest number of jumps needed to go from the top left corner to the bottom right corner, given the layout of the launchers on the board.

Input format

The first line of the input contains two positive integers M and N, giving the dimensions of the chessboard. M is the number of rows of the board and N is the number of columns. This is followed by $2M$ lines of input: M lines describing the R values of the launchers followed by M lines describing the D values of the launchers. Line $1+i$, $1 \leq i \leq M$, has N integers, describing the R values for row i. Line $M+1+i$, $1 \leq i \leq M$, has N integers, describing the D values for row i.

Output format

The output should be a single integer, the minimum number of jumps required to reach the bottom right square from the top left square on the given chessboard.

Test data

You may assume that $1 \leq M \leq 250$ and $1 \leq N \leq 250$.

Example

Here is the sample input and output corresponding to the example discussed above.

Sample input

```
3 4
1 1 1 2
3 1 1 1
1 1 1 2
2 2 2 1
1 1 2 2
1 1 2 2
```

Sample output

```
3
```

Note: Your program should not print anything other than what is specified in the output format. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Important

Indicate the location of the source code file for your solution to Question 1 in the boxes below. *If you fail to do this, your solution cannot be evaluated!*

Source file:

Question 2 The Gangster's Authentication Protocol

How can two gangsters who have never met before identify each other when they meet to exchange stolen goods? One technique that we see in movies is to tear a photograph or a currency note into two and send one piece with each of them. At the meeting the two gangsters can verify each other's identity by matching the two pieces.

What if K gangsters, $K > 2$, were to meet? We could tear up the photograph or currency note into K pieces and send a piece with each of them. But then, when the gangsters meet, they have to spend a long time solving a rather difficult jigsaw puzzle. In underworld meetings, speed is of the essence, to avoid attracting the attention of the police.

The boss of one gang has come up with an intelligent plan that uses numbers rather than photographs or currency notes. He picks a number $N > K$. He then partitions the set $\{1, 2, \ldots, N\}$ into K nonempty collections and assigns one collection to each gangster who is part of the meeting. When the K gangsters meet, all that they have to verify is that, between them, they have all the numbers between 1 and N with no repetitions.

To make things more secure, the boss decides that he will not directly give each gangster his collection. Instead, he plans to encode the collections as follows. Each element m in a collection is replaced by the number of elements in the *other* collections that are less than m.

For example, suppose that $K = 3$, $N = 10$ and the collections assigned to the three gangsters are $\{5, 2, 9\}$, $\{1, 8, 7\}$ and $\{10, 3, 6, 4\}$. Then, the coded collections given to these three gangsters would be $\{3, 1, 6\}$, $\{0, 5, 5\}$ and $\{6, 2, 3, 2\}$, respectively. In the first collection, 5 is replaced by 3, since there are three elements outside this collection, $\{1, 3, 4\}$, that are less than 5. The number 2 is replaced by 1 since there is one element outside this collection, $\{1\}$, which is less than 2. Finally, 9 is replaced by 6, since there are 6 elements outside this collection, $\{1, 3, 4, 6, 7, 8\}$, that are less than 9. You can check that the other two coded collections are also obtained in the same manner.

Unfortunately, the boss is still struggling to come up with a way for the K participants to look at the K encoded collections that they have with them when they meet and reconstruct the original collections assigned to each of them. Without this, they cannot verify that every number between 1 and N appears in exactly one of these collections, which they need to do to establish that there are no impostors in the meeting.

Your task is help the boss by writing a program that will look at all the coded collections assigned to the gangsters and reconstruct the actual collections assigned to each of them. You can assume that the input to your program always describes a valid set of coded collections—that is, a set for which the corresponding decoded collections constitute a partition of $\{1, 2, \ldots, N\}$.

Input format

The first line of the input contains two positive integers N and K, with $N > K$. The following K lines contain a description of the coded sets assigned to the K gangsters. Line $1+i$, $1 \leq i \leq K$, describes the coded set assigned to gangster i. The first integer on line $1+i$,

M_i, gives the number of elements in the set assigned to gangster i. This is followed by M_i positive integers, the elements of the coded set assigned to gangster i.

Output format

The output should have K lines. Line i should list out the actual decoded elements of the set assigned to gangster i. These elements may be listed out in any order.

Test data

You may assume that $1 \leq N \leq 2000$ and $1 \leq K \leq 2000$.

Example

Here is the sample input and output corresponding to the example discussed above.

Sample input

```
10 3
3 3 1 6
3 0 5 5
4 6 2 3 2
```

Sample output

```
9 5 2
1 8 7
3 4 6 10
```

Note: Your program should not print anything other than what is specified in the output format. Please remove all diagnostic print statements before making your final submission. A program with extraneous output will be treated as incorrect!

Important

Indicate the location of the source code file for your solution to Question 2 in the boxes below. *If you fail to do this, your solution cannot be evaluated!*

Source file: