LAB-PRAC-09_PTR-MAT

Mr C writes a Story (p1v1d1)

Mr C writes a Story [10 marks]

Problem Statement
Mr C is trying to write a story. However, he is so tired from ESC101 duties that all is not going according to plan. He starts writing the story, writes a few characters, but falls asleep in the middle and after that types some garbage characters. He wakes up, all groggy, and being unable to distinguish story from garbage, starts writing the story all over again. However, he yet again falls asleep midway and this process repeats a few times.

In your input, the first line will contain a single strictly positive integer n indicating the number of attempts Mr C made at writing his story. n will be no more than 100. The next n lines will each contain a single attempt. The attempts may contain printable characters, spaces, tabs but no non-printable characters. Each attempt will be contained in a single line. Each attempt will contain no more than 499 characters. We also promise that no attempt will contain zero characters.

You have to print the longest *prefix substring* that is common to all attempts of Mr C in you output. This is arguably the longest portion of the story Mr C was ever able to complete in an attempt before falling asleep. A prefix substring is a substring that begins with the first character of the string e.g. "P", "Pr", "Pre", "Prea" are all prefix substrings of "Preamble" but the substrings "amble", "l", "eam", are not valid prefix substrings. For this question, the empty substring with zero characters is considered a valid prefix substring of any string.

If there is no prefix substring common to all attempts of Mr C (in other words if the empty substring is the only common substring) then print "EMPTY" in your output.

Caution

1. The prefix substring can contain spaces and tabs.
2. Note n is going to be provided in the first line and the strings on line 2 onward. This means there will be a new line character after n. Be careful not to read that newline character into your strings.
3. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE 1:
INPUT
3
TheSongOfIce
TheSongOfFire
TheDanceOfWinds

OUTPUT:
The

EXAMPLE 2:
INPUT

3
Wake up
Fall down
Rise up

OUTPUT:
EMPTY

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
3 The Song Of Ice The Song Of Fire The Silence of the Lambs	The S
4 Wake up Fall down Rise up Do it again	EMPTY
6 I will write my story today I will wzzzzzzz I will zzzz my story today I wzzzzzzz I will write mzzzzzzzz I will write my story today	I w
6 I will write my story today I will wzzzzzzz I will zzzz my story today I wzzzzzzz I will write mzzzzzzzz I will write my story today	EMPTY
1 I will definitely write my story today	I will definitely write my story today
15 abcdefghijklmnopqrstuvwxyz abcdefghijklmnopqrstuvw abcdefghijklmnopqrst abcdefghijklmno abcdefghijkl abcdefghi abcdef abc abcdef	abc

abcdefghi abcdefghijkl abcdefghijklmno abcdefghijklmnopqrst abcdefghijklmnopqrstuvw abcdefghijklmnopqrstuvwxyz	

Matrix Arithmetic (p1v2d1)

Matrix Arithmetic [20 marks]

Problem Statement
Matrices possess a lot of the structures integers possess, some of which will be introduced to you in MTH102 and beyond. In this question we will look at two simple operations on squares matrices. In the first line of the input you will be given 2 strictly positive integers m and p. In the next m lines you will be given the entries of an m x m square integer matrix - lets call this matrix A. Each row of the matrix will be given in a different line with two elements separated by a single space.

You will have to output the matrices $P = p \cdot A$ and $Q = A^p$ in your output. The matrix $p \cdot A$ is defined as the matrix each of whose entries is equal to p multiplied with the corresponding entry of A i.e. i,j entry of P = i,j entry of A times p. The matrix A^p is obtained as the iterated product (((((A \cdot A) \cdot A) \cdot A) \cdot A) ... \cdot A) where A is multiplied with itself p times (since A is square, these products make sense dimensions wise).

Print the matrices P and Q with each row printed on a different line with a single space between two elements of each row. There should be no trailing whitespaces at the end of any line. n and p will be smaller than or equal to 100 and all your outputs will fit inside the int datatype.

Caution

1. Although m, p will always be strictly positive, the matrix entries may be zero or negative as well.
2. Matrix product is not commutative but it is associative i.e. if the product of three matrices A, B, C makes sense (i.e. dimensions wise) then we always have (A * B) * C = A * (B * C)
3. Be careful about extra/missing lines and extra/missing spaces in your output.

HINTS:

1. You may try to compute the matrix exponential as a *running power* i.e. find A^1 and then A^2 and then A^3 and so on.
2. In order to find the matrix exponential A^p, you have to organize your storage a bit. Use three matrices, one to store A, another to store the running power, and a third temporary matrix. Use the temporary matrix to compute and store the next power of A, given the previous power of A and then copy these values to the running power matrix.

Code to manipulate matrices

```
int num[4][6] = {
{1,2,3,4,5,6},
```

```
{10,20,30,40,50,60},
{100,200,300,400,500,600},
{1000,2000,3000,4000,5000,6000}
};
int i,j;
for(i = 0; i < 4; i++){
for(j = 0; j < 6; j++){
printf("%d", num[i][j]);
if(j < 5) printf(" "); // No stray spaces at the end
}
if(i < 3) printf("\n"); // No stray new lines at the end
}
```

EXAMPLE:
INPUT
2 2
1 2
2 1

OUTPUT:
2 4
4 2
5 4
4 5

Explanation: The product matrix P is
2 4
4 2
and the matrix exponential is
5 4
4 5

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are will be 2 * m lines in your output, depending on the value of m. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 2 1 2 2 1	2 4 4 2 5 4 4 5
4 5 1 1 1 1 1 1 1 1	5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

1 1 1 1 1 1 1 1	256 256 256 256 256 256 256 256 256 256 256 256 256 256 256 256
	-30 0 0 0 0 0 30 0 0 0 0 0 -30 0 0 0 0 0 30 0 0 0 0 0 -30
5 30 -1 0 0 0 0 0 1 0 0 0 0 0 -1 0 0 0 0 0 1 0 0 0 0 0 -1	1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1
7 10 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1024 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
9 100 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1	100 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 100 0 0 0 0 0 0 0 0 0 100 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
4 5 1 1 1 1 0 1 1 1 0 0 1 1 0 0 0 1	5 5 5 5 0 5 5 5 0 0 5 5 0 0 0 5 1 5 15 35 0 1 5 15 0 0 1 5 0 0 0 1

Spin the Matrix (p1v3d1)

Spin the Matrix [20 marks]

Problem Statement
In the first line you will be given a single strictly positive integer n. In the next n lines, you will be given an n x n matrix, call it A, filled with integers. Each row of the matrix will be given in a single line with entries in a row separated by a single space. In the last (n+2)-th line of the input, you will be given a list of characters delimited by the character 'X'. The list will only contain the characters 'L' and 'R' and not contain the terminating character 'X'. The list will contain at least one 'L' or 'R' character but no more than 99 L/R characters.

Each 'L' is an instruction to *rotate* the matrix A counterclockwise by 90 degrees (see example below). Each 'R' is an instruction to *rotate* the matrix A clockwise by 90 degrees (see example below). Find the matrix that forms after following the L/R instructions in the list. Call this matrix B.

For your output, first print the matrix as you would get it if you were to just rotate it 90 degrees clockwise. Then output the matrix B as calculated above (by following the instructions in the list).

Caution

1. When printing matrices, print each row on a separate line, with a single space between two elements. Be careful to not include any stray spaces at the end of any line.
2. There will be a newline character after n as well as one before the list of L/R begins. Be careful to discard these newline characters.
3. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE:
INPUT
2
1 2
3 4
LRLX

OUTPUT:
3 1
4 2
2 4
1 3

Explanation: After rotating the matrix A once clockwise 90 degrees, we get the matrix
3 1
4 2
If we are to follow the instructions LRL, we effectively rotate the matrix counterclockwise 90 degrees (L and R cancel each other) and we get the matrix
2 4
1 3

Grading Scheme:

Total marks: **[20 Points]**

There will be partial grading in this question. There are 2 * n in your output depending on n. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 1 2 3 4 LRLX	3 1 4 2 2 4 1 3
3 1 1 1 0 1 1 0 0 1 LRRRRRLLRX	0 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 0
5 1 2 3 4 5 0 2 3 4 5 0 0 3 4 5 0 0 0 4 5 0 0 0 0 5 LRRRRLLRX	0 0 0 0 1 0 0 0 2 2 0 0 3 3 3 0 4 4 4 4 5 5 5 5 5 5 0 0 0 0 5 4 0 0 0 5 4 3 0 0 5 4 3 2 0 5 4 3 2 1
4 1 2 3 4 0 2 3 4 0 0 3 4 0 0 0 4 LLLLRRRRLRX	0 0 0 1 0 0 2 2 0 3 3 3 4 4 4 4 1 2 3 4 0 2 3 4 0 0 3 4 0 0 0 4
6 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 LLLX	0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
7 8 8 8 8 8 8 8	0 8 8 8 8 8 8 8 8 8 8 8 8 8

8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 0 8 8 8 8 8 8 RLLLRLRRRX	8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 0 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8

Crony Capitalization (p2v1d1)

Crony Capitalization [10 marks]

Problem Statement
Mr C has a friend who likes to type in a very untidy manner and then cajoles Mr C to clean up his text. The first line of the input will contain a strictly positive integer n followed by n lines, each containing an untidy string. Each string will contain English alphabets (upper and lower case), numerals, spaces as well as all punctuation marks (brackets, colon, semicolon, quotes etc). The three punctuation marks full stop . exclamation mark ! and question mark ? will be designated as *termination symbols*.

Your output should convert each of the n input lines and convert them to proper capitalization as described below. Print each corrected line in a separate output line. The rules for for capitalization are very simple

1. Sentences are terminated by termination symbols and may be followed by one or more spaces before the next sentence starts. Each line of input will contain one or more sentences.
2. Sentences must contain exactly one termination symbol but may otherwise be empty.
3. If the first non-space character in a sentence is an alphabet character, it must be in upper case.
4. All other alphabet characters in a sentence must be in lower case.
5. Case rules do not apply to non-alphabet characters.

Caution

1. Sentences may start with any non-space character.
2. Sentences may be empty.
3. Your output lines must not differ from the input lines in terms of words or their order or number or spaces etc. The only modification that is needed is fixing the capitalization.
4. Your output must contain exactly n lines.
5. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE:
INPUT
2

Hello, My name is Mr C
What is your name? I teach ESC101

OUTPUT:
Hello, my name is mr c
What is your name? I teach esc101

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
2 Hello, My name is Mr C What is your name? I teach ESC101	Hello, my name is mr c What is your name? I teach esc101
3 i'M trYinG to Stop tYping Like this. but it's not heLPINg. senD helP! this IS CATASTROPHIC!	I'm trying to stop typing like this. But it's not helping. Send help! This is catastrophic!
2 Welcome to ESC101! 450 students in single class is a lot :) wouldn't you think? I think so.	Welcome to esc101! 450 students in single class is a lot :) wouldn't you think? I think so.
5 i'M trYinG to Stop tYping Like this. but it's not heLPINg. senD helP! this IS CATASTROPHIC!	I'm trying to Stop typing like this. But it's not helping. Send help! This is catastrophic!
5 i'M trYinG to Stop tYping Like this. but it's not heLPINg. senD helP! this IS CATASTROPHIC!	I'm trying to Stop typing like this. But it's not helping. Send help! This is catastrophic!
2 Thanks! i appreciate your help... see you soon. well, i am glad you got the help you needed :) you are welcome	Thanks! I appreciate your help... See you soon. Well, i am glad you got the help you needed :) you are welcome

---

Matrix Mirroring (p2v2d1)

Matrix Mirroring [20 marks]

Problem Statement
In the first line of the input you will be given two strictly positive integers n and m. In the next n lines you will be given the n rows of an n x m integer matrix, one row in each line, with a single space separating two entries in a row. You have to perform two mirroring operations on the matrix which will enlarge the matrix, and print the final output. See below for an example.

Caution

1. Print each row of your output matrix on a separate line. There should be a single space between two entries in a row.
2. Be careful not to have any trailing spaces at the end of any line or else have any trailing new lines at the end of the output.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
printf("%d", num[i][j]);
```

EXAMPLE:
INPUT
2 2
10 20
30 40

Explanation First mirror the matrix vertically to get the following enlarged matrix
10 20
30 40
30 40
10 20

Now mirror this enlarged matrix horizontally to get an even larger matrix.
10 20 20 10
30 40 40 30
30 40 40 30
10 20 20 10

OUTPUT:
10 20 20 10
30 40 40 30
30 40 40 30
10 20 20 10

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line

correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 2 10 20 30 40	10 20 20 10 30 40 40 30 30 40 40 30 10 20 20 10
2 3 1 2 3 4 5 6	1 2 3 3 2 1 4 5 6 6 5 4 4 5 6 6 5 4 1 2 3 3 2 1
1 1 8	8 8 8 8
5 1 100 200 300 400 500	100 100 200 200 300 300 400 400 500 500 500 500 400 400 300 300 200 200 100 100
1 5 1 11 111 1111 11111	1 11 111 1111 11111 11111 1111 111 11 1 1 11 111 1111 11111 11111 1111 111 11 1
10 10 9 1 8 6 8 2 4 8 5 9 7 5 4 8 6 2 2 3 4 9 6 3 4 5 7 8 1 2 6 7 0 8 7 6 5 4 9 8 7 6 2 5 6 1 5 4 8 4 6 8 7 5 4 8 6 2 2 3 4 9 6 3 4 5 7 8 1 2 6 7 9 1 8 6 8 2 4 8 5 9 7 5 4 8 6 2 2 3 4 9 6 3 4 5 7 8 1 2 6 7	9 1 8 6 8 2 4 8 5 9 9 5 8 4 2 8 6 8 1 9 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 0 8 7 6 5 4 9 8 7 6 6 7 8 9 4 5 6 7 8 0 2 5 6 1 5 4 8 4 6 8 8 6 4 8 4 5 1 6 5 2 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 9 1 8 6 8 2 4 8 5 9 9 5 8 4 2 8 6 8 1 9 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 9 1 8 6 8 2 4 8 5 9 9 5 8 4 2 8 6 8 1 9 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 2 5 6 1 5 4 8 4 6 8 8 6 4 8 4 5 1 6 5 2 0 8 7 6 5 4 9 8 7 6 6 7 8 9 4 5 6 7 8 0 6 3 4 5 7 8 1 2 6 7 7 6 2 1 8 7 5 4 3 6 7 5 4 8 6 2 2 3 4 9 9 4 3 2 2 6 8 4 5 7 9 1 8 6 8 2 4 8 5 9 9 5 8 4 2 8 6 8 1 9

Sodoku (p2v3d1)

Sodoku [20 marks]

Problem Statement
The Sodoku is a popular puzzle, commonly played in its 9 x 9 version. We will look at a generalization of this puzzle. An n x n Sodoku is defined whenever n is a perfect square like 4 or 9 or 16 etc and is presented as an n x n matrix, each of whose entries is an integer between 1 and n.

Let k be the square root of n. Then for an n x n matrix, we define n non-overlapping *boxes* as follows. Consider the following example 4 x 4 Sodoku.
1 2 3 4
3 4 1 2
4 3 2 1
2 1 4 3

Here, the 4 boxes are
BOX 1
1 2
3 4

BOX 2
3 4
1 2

BOX 3
4 3
2 1

BOX 4
2 1
4 3

i.e. the boxes are divided as

1 2 || 3 4
3 4 || 1 2
=========
4 3 || 2 1
2 1 || 4 3

Note that boxes are non-overlapping and are numbered left to right and top to bottom. There are exactly n boxes in the matrix. A Sodoku is considered valid if

1. Every row of the matrix has all numbers from 1 to n occurring exactly once.
2. Every column of the matrix has all numbers from 1 to n occurring exactly once.
3. Every box in the matrix has all numbers from 1 to n occurring exactly once.

You will be given n as a strictly positive perfect square integer in the first line of the input. Then you will be given the n rows of this matrix, each row on a separate line, with a single space separating two elements of a row. If the given Sodoku is valid, simply print "Valid Sudoku" (without quotes) in the output and that is it.

However, if the Sodoku is not valid, you have to first described which all rows are invalid (in increasing order of rows) then describe which all columns are invalid (in increasing order of columns) then describe which all boxes are invalid (in increasing order of boxes).

Caution

1. We will not penalize you for trailing new lines at the end of your output. However, do not have trailing spaces at the end of any line of your output.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
printf("%d", num[i][j]);
```

HINT: You may use the sqrt function by including math.h to calculate the square root of a number.

EXAMPLE 1:

INPUT
4
1 2 3 4
3 4 1 2
4 3 2 1
2 1 4 3

OUTPUT:
Valid Sudoku

EXAMPLE 2:

INPUT
4
1 2 3 2
3 4 1 2
4 3 2 1
2 1 4 3

OUTPUT:
Row 1 is invalid
Column 4 is invalid
Box 2 is invalid

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There will be several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
4 1 2 3 2 3 4 1 2 4 3 2 1 2 1 4 3	Row 1 is invalid Column 4 is invalid Box 2 is invalid
1 2	Row 1 is invalid Column 1 is invalid Box 1 is invalid
1 1	Valid Sudoku
9 2 4 3 8 7 2 9 5 1 9 5 1 2 4 3 8 6 7 8 7 6 1 9 5 2 4 3 3 9 4 5 2 8 1 7 6 1 6 2 4 3 7 5 8 9 7 8 5 7 6 1 3 2 4 6 2 9 3 8 4 7 1 5 3 3 7 6 1 2 4 9 8 4 1 8 7 5 9 6 3 1	Row 1 is invalid Row 6 is invalid Row 8 is invalid Row 9 is invalid Column 1 is invalid Column 4 is invalid Column 6 is invalid Column 9 is invalid Box 2 is invalid Box 5 is invalid Box 7 is invalid Box 9 is invalid
16 12 10 12 9 2 6 15 5 4 3 7 11 14 16 13 1 1 15 5 2 1 8 14 7 9 13 10 16 11 12 6 3 11 14 16 7 13 3 9 12 1 6 2 5 4 11 10 15 4 13 6 3 11 1 10 16 14 8 15 12 2 5 7 9 13 5 7 11 12 9 3 2 6 15 1 8 10 4 16 14 9 3 15 9 8 13 1 11 7 10 16 14 12 6 2 9 10 16 1 8 5 15 6 14 13 4 10 2 3 11 9 7 6 2 12 14 7 16 4 10 3 5 11 9 13 15 1 8 15 9 10 16 3 2 5 8 15 11 13 4 7 1 14 6 5 8 14 12 6 4 7 13 2 9 3 1 16 10 15 11 3 4 11 1 10 14 16 15 5 7 8 6 9 2 12 13 7 6 2 13 9 11 12 1 10 16 14 15 8 3 5 4 8 11 9 10 14 12 2 3 15 1 6 13 5 7 4 16 14 12 13 15 16 5 8 6 11 2 4 7 1 9 3 10 2 1 3 5 15 7 11 4 16 14 9 10 6 13 8 12 16 7 4 6 1 10 13 9 8 12 5 3 15 14 11 2	Row 1 is invalid Row 2 is invalid Row 3 is invalid Row 6 is invalid Row 7 is invalid Row 9 is invalid Column 3 is invalid Column 4 is invalid Column 5 is invalid Column 9 is invalid Column 11 is invalid Column 14 is invalid Column 16 is invalid Box 1 is invalid Box 2 is invalid Box 4 is invalid Box 5 is invalid Box 7 is invalid Box 8 is invalid Box 11 is invalid
4 4 5 2 3 1 2 3 4 2 3 4 1 3 4 1 2	Row 1 is invalid Column 2 is invalid Box 1 is invalid Box 2 is invalid Box 3 is invalid Box 4 is invalid

The Last Line (p3v1d1)

The Last Line [10 marks]

Problem Statement
Mr C likes debating with his friend Mr B. However, Mr B tends to make a lot of long arguments and usually, only the last sentence he speaks is worth taking into account. In your input, you will be given a speech Mr B gave. You will have to extract the last sentence he spoke. Your input will consist only of upper and lower case English alphabets, digits 0-9, comma , termination symbols (fullstop . question mark ? exclamation mark), spaces and newlines.

A sentence will always be terminated by a termination symbol (. or ? or !) and the next sentence will always begin after one or more spaces are given after the termination symbol. The speech will contain no more than 999 characters in total and we promise that the last character will always be a termination symbol (never a space or an English alphabet).

Caution

1. The speech may span multiple lines of input
2. The last sentence may itself span multiple lines of input
3. There may be only one sentence in the entire speech which by default becomes the last sentence.
4. Your output should always start with a non-space character (sentences never start with a space).
5. Be careful about extra/missing lines and extra/missing spaces in your output.

EXAMPLE 1:
INPUT
Hello! How are you today?

OUTPUT:
How are you today?

EXAMPLE 2:
INPUT
Hello! How are you
today?

OUTPUT:
How are you
today?

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output

Hello ESC 101! We hope you enjoy this lab.	We hope you enjoy this lab.
Pointers are a lot of fun. However, we need to be careful!	However, we need to be careful!
Pointers are a lot of fun. However, do we need to be careful?	However, do we need to be careful?
Well, pointers are a form of low level programming. They are quite common! Many languages support them!	Many languages support them!
ESC . 101 . is . a . lot . of . fun!	fun!
This is the last sentence!	This is the last sentence!

Singular Value Decomposition (p3v2d1)

Singular Value Decomposition [20 marks]

Problem Statement
Every m x n real matrix A can be decomposed as A = U * S * V' (where ' indicates the transpose operation) where U is an m x m unitary matrix (i.e. U is its own inverse), S is an m x n diagonal matrix (all off-diagonal entries of S, i.e. entries of the form S_{ij} where $i \neq j$ are zero. Only entries of the form S_{ii} can be non-zero) and V is an n x n unitary matrix (i.e. V is its own inverse).

Both m and n are guaranteed to be less than or equal to 100. In the first line of the input, you will be given m and n as two strictly positive integers, separated by a single space. In the next line, you will be given the diagonal entries of S in a single line (two entries separated by a space). In then next m lines, you will be given the m rows of the matrix U and in the next n lines you will be given the n rows of the matrix V. U, S, V will have only integer entries.

You have to output the matrix A = U * S * V' as your output. Output one row of A in each line with a single space between two entries of that row. Make sure there are no trailing spaces at the end of each line, as well as no trailing new lines in your output.

Caution

1. Be careful about extra/missing lines and extra/missing spaces in your output.
2. V appears transposed in the expression for A
3. Be careful to count how many diagonal entries, a rectangular matrix has.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
printf("%d", num[i][j]);
```

EXAMPLE:
INPUT
3 2
2 2
1 0 0
0 1 0
0 0 1
1 0
0 1

OUTPUT:
2 0
0 2
0 0

Explanation: In this case, the matrix S will look like
2 0
0 2
0 0
Note that only diagonal entries are non-zero.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There will be m lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
3 2 2 2 1 0 0 0 1 0 0 0 1 1 0 0 1	2 0 0 2 0 0
3 3 6 7 0 1 2 3 0 1 2 0 0 1 1 0 9 5 1 7 2 8 1	6 44 124 0 7 56 0 0 0
2 4 1 1 1 0	1 5 2 3 4 21 16 20

4 1 1 0 9 1 5 1 7 1 2 8 1 1 3 8 9 5	
2 1 5 1 2 3 4 6	30 90
1 6 10 5 1 2 3 4 5 6 6 5 4 3 2 1 1 2 3 4 5 6 6 5 4 3 2 1 1 2 3 4 5 6 6 5 4 3 2 1	50 300 50 300 50 300
3 4 0 0 0 1 2 3 4 5 6 7 8 9 1 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0

Matrix Flip (p3v3d1)

Matrix Flip [20 marks]

Problem Statement
In the first line of the input, you will be given two strictly positive integers n and m, separated by a space. In the next n lines, you will be given an n x m matrix A of digits (non-negative single digit integers) with each line containing a single row of the matrix. Two entries of the matrix will be separated by a single space. In the last line of the input, you will be given a list of characters which will all be capital (upper case) English alphabet letters. The list will be terminated by the character 'X' which will appear only once in the entire list, at the end. The list will be non-empty i.e. the first character in the list will not be X.

You have to interpret the characters as instructions. If the character is 'H' you have to flip the matrix A horizontally, print "HORIZONTAL" on the output followed by a newline followed by the matrix A after the flip. If the character is 'V' you have to flip the matrix A vertically, print "VERTICAL" on the output followed by a newline followed by the matrix A after the flip. If the character is neither H nor V nor X, print "ILLEGAL" on the output followed by a newline followed by the matrix as is it is at that point.

Caution

1. Note that the instructions given to you have to be followed in sequence. This means that if the sequence is HVUH then flip the matrix horizontally, (print the word HORIZONTAL followed by the

matrix) then vertically (print the word VERTICAL followed by the matrix) then print the word ILLEGAL followed by the matrix (do nothing to the matrix), then flip horizontally (print the word HORIZONTAL followed by the matrix).
2. While printing the matrix A, print each row of the matrix on a separate line with a single space between two elements of a row.
3. If you are using getchar to read in characters in the last line, make sure that you do not read in by mistake a trailing newline character left behind from the previous line.
4. Make sure there are no trailing spaces at the end of each line and no trailing newlines.
5. Be careful about extra/missing lines and extra/missing spaces in your output.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
printf("%d", num[i][j]);
```

EXAMPLE:

INPUT
2 3
1 2 3
4 5 6
HVX

OUTPUT:
HORIZONTAL
3 2 1
6 5 4
VERTICAL
6 5 4
3 2 1

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are a certain number of lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 3 1 2 3 4 5 6 HVDX	HORIZONTAL 3 2 1 6 5 4 VERTICAL

	6 5 4 3 2 1 ILLEGAL 6 5 4 3 2 1
1 4 1 2 3 4 HHHVX	HORIZONTAL 4 3 2 1 HORIZONTAL 1 2 3 4 HORIZONTAL 4 3 2 1 VERTICAL 4 3 2 1
4 1 1 2 3 4 VVHHX	VERTICAL 4 3 2 1 VERTICAL 1 2 3 4 HORIZONTAL 1 2 3 4 HORIZONTAL 1 2 3 4
2 4 0 1 2 3 9 8 7 6 HVWWVHX	HORIZONTAL 3 2 1 0 6 7 8 9 VERTICAL 6 7 8 9 3 2 1 0 ILLEGAL 6 7 8 9 3 2 1 0 ILLEGAL 6 7 8 9 3 2 1 0 VERTICAL 3 2 1 0 6 7 8 9 HORIZONTAL 0 1 2 3 9 8 7 6
3 5 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 HVHVDVHHVX	HORIZONTAL 4 3 2 1 0 5 4 3 2 1 6 5 4 3 2 VERTICAL

	6 5 4 3 2 5 4 3 2 1 4 3 2 1 0 HORIZONTAL 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 VERTICAL 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 ILLEGAL 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 VERTICAL 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 HORIZONTAL 6 5 4 3 2 5 4 3 2 1 4 3 2 1 0 HORIZONTAL 2 3 4 5 6 1 2 3 4 5 0 1 2 3 4 VERTICAL 0 1 2 3 4 1 2 3 4 5 2 3 4 5 6
4 6 0 1 2 9 8 7 1 2 3 8 7 6 2 3 4 7 6 5 3 4 5 6 5 4 HVGHVDVHHTVX	HORIZONTAL 7 8 9 2 1 0 6 7 8 3 2 1 5 6 7 4 3 2 4 5 6 5 4 3 VERTICAL 4 5 6 5 4 3 5 6 7 4 3 2 6 7 8 3 2 1 7 8 9 2 1 0 ILLEGAL 4 5 6 5 4 3 5 6 7 4 3 2 6 7 8 3 2 1 7 8 9 2 1 0 HORIZONTAL 3 4 5 6 5 4 2 3 4 7 6 5 1 2 3 8 7 6 0 1 2 9 8 7 VERTICAL 0 1 2 9 8 7 1 2 3 8 7 6 2 3 4 7 6 5 3 4 5 6 5 4 ILLEGAL

	0 1 2 9 8 7 1 2 3 8 7 6 2 3 4 7 6 5 3 4 5 6 5 4 VERTICAL 3 4 5 6 5 4 2 3 4 7 6 5 1 2 3 8 7 6 0 1 2 9 8 7 HORIZONTAL 4 5 6 5 4 3 5 6 7 4 3 2 6 7 8 3 2 1 7 8 9 2 1 0 HORIZONTAL 3 4 5 6 5 4 2 3 4 7 6 5 1 2 3 8 7 6 0 1 2 9 8 7 ILLEGAL 3 4 5 6 5 4 2 3 4 7 6 5 1 2 3 8 7 6 0 1 2 9 8 7 VERTICAL 0 1 2 9 8 7 1 2 3 8 7 6 2 3 4 7 6 5 3 4 5 6 5 4

Now we are in Rome (p4v1d1)

Now we are in Rome [10 marks]

Problem Statement

We learnt the binary, octal, hexadecimal and decimal number systems in the lectures. Letâ€™s go back in the history. Before the decimal system and other place value systems were introduced, other number systems were popular. One among them (which still finds place among people) is the Roman number system. The Roman numeral system is based on seven symbols and calculates values based on three simple rules (addition, subtraction, and concatenation rule). The numbers are represented using a sequence of these seven symbols.

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Addition Rule: When read from right to left, till the characters appear in non-decreasing order of their value, their values keep getting added to the final total value. For example III is 3, MX is 1010, CLXI is 161, XX is 20.

III	3	1 + 1 + 1
CLXI	161	100 + 50 + 10 + 1
MMCXVIII	2118	2*1000 + 100 + 10 + 5 + 3*1

Subtraction Rule: If when read from right to left, if a character with lower value appears after a character of higher value, then the value of the lower value character is subtracted from the total i.e. IV is 4 and IX is 9, CM is 900.

XLIX	49	(-10) + 50 + (-1) + 10
CMXCIX	999	(-100) + 1000 + (-10) + (100) + (-1) + 10
MMCXIX	2019	2*1000 + 100 + 10 + (-1) + 10

Concatenation Rule: Once subtraction rule has been applied, the characters must again resume being of higher and higher value so that addition rule kicks in again. This means that a number like IVX is illegal.

In your input you will be given a roman numeral as a string of no more than 15 characters. Find the corresponding decimal value and print it in the output.

Note: We must admit that we have not explained all the rules of the roman numeral system here since explaining all the rules would take ages. The system is actually very cumbersome and not neat at all which is why people stopped using it the moment the decimal system was available. However, believe it or not, some people were actually afraid of the decimal system and thought the decimal system was the work of the devil. In hindsight it is the Roman system that seems like a devilishly complicated system :)

EXAMPLE:
INPUT
XIV

OUTPUT:
14

Grading Scheme:
Total marks: **[10 Points]**

There will be no partial grading in this question. An exact match will receive full marks whereas an incomplete match will receive 0 points. Please be careful of missing/extra spaces and missing/lines (take help of visible test cases). Each visible test case is worth 1 point and each hidden test case is worth 2 points. There are 2 visible and 4 hidden test cases.

All Test Cases (Visible + Hidden)

Input	Output
MMXVIII	2018
XLIX	49
MMXIX	2019
MCMXCIX	1999

DCLXVI	666
LXXXIX	89

Search for the Submatrix (p4v2d1)

Search for the Submatrix [20 marks]

Problem Statement
Given a string, any contiguous set of characters occurring in that string is considered a substring of that string. Similarly we can extend that notion to submatrices as well. In the first line of the input, you will be given two strictly positive integers n and m, separated by a space. In the next n lines, you will be given the entries of an n x m integer matrix A with one row of A in each line with two entries separated by a single space.

Then in the next line you will be given two strictly positive integers k and l, separated by a space. In the next k lines, you will be given the entries of a k x l integer matrix B with one row of B in each line with two entries separated by a single space. We promise that k will be less than or equal to n and l will be less than or equal to m.

In your output you have to tell us if the matrix B occurs as a submatrix in A or not. If B never occurs as a submatrix inside A, simply print the words "SUBMATRIX NOT PRESENT" (without quotes) in the output. However, if B is present one or more times in A as a submatrix, then you have to print the indices (as in 2D array indices) of the top left hand corners of all these occurrences in the output, one occurrence in each line in a format given below.

Make sure you first output all occurrences (if any) where the top left hand corner is in the first row of A, followed by all occurrences (if any) where the top left hand corner is in the second row of A, followed by all occurrences (if any) where the top left hand corner is in the third row of A and so on and so forth. If there are multiple occurrences with top left hand corner on the same row of A, output these occurrences in increasing order of the column number (index) of the top left hand corner.

Caution

1. We will not penalize you if there are extra newlines at the end of your output.
2. However, there should not be any stray, trailing spaces in your output.
3. Note that if your occurrences are given in the wrong order, you may end up getting zero by the autograder even if you output all occurrences correctly. This is because in order to get partial credit (see grading scheme below), you must output the correct occurrence in the correct order.
4. Submatrix occurrences may overlap with each other, just as substrings may overlap with each other.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
printf("%d", num[i][j]);
```

EXAMPLE:
INPUT
2 4
1 5 1 5
2 6 1 5
1 2
1 5

OUTPUT:
(0, 0)
(0, 2)
(1, 2)

Explanation: Note that there were three occurrences of B in A but the two occurrences in the first row got reported first and then the occurrence in the second row got reported. Also, within the two occurrences in the first row, the one with smaller column number got reported first.

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There may be several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 4 1 5 1 5 2 6 1 5 1 2 1 5	(0, 0) (0, 2) (1, 2)
2 3 1 5 10 2 6 11 1 2 1 6	SUBMATRIX NOT PRESENT
4 10 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	(0, 0) (0, 1) (0, 2) (0, 3) (0, 4) (0, 5) (0, 6) (0, 7) (0, 8) (0, 9) (1, 0) (1, 1) (1, 2) (1, 3)

	(1, 4) (1, 5) (1, 6) (1, 7) (1, 8) (1, 9) (2, 0) (2, 1) (2, 2) (2, 3) (2, 4) (2, 5) (2, 6) (2, 7) (2, 8) (2, 9) (3, 0) (3, 1) (3, 2) (3, 3) (3, 4) (3, 5) (3, 6) (3, 7) (3, 8) (3, 9)
3 7 1 2 1 2 1 2 1 0 1 2 1 2 1 0 0 0 1 2 1 0 0 1 3 1 2 1	(0, 0) (0, 2) (0, 4) (1, 1) (1, 3) (2, 2)
7 6 1 0 0 1 0 0 2 1 0 2 1 0 1 2 1 1 2 1 2 1 2 2 1 2 1 2 1 1 2 1 2 1 0 2 1 0 1 0 0 1 0 0 3 1 1 2 1	(0, 0) (0, 3) (1, 1) (1, 4) (2, 0) (2, 2) (2, 3) (2, 5) (3, 1) (3, 4) (4, 0) (4, 3)
5 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 0 0 1 2 3 4 5 0 0 6 7 8 9 0 0 0 5 4 3 2 1 3 5 1 2 3 4 5 6 7 8 9 0 5 4 3 2 1	(2, 2)

Convoluted Convolutions (p4v3d1)

Convoluted Convolutions [20 marks]

The process of convolution is immensely useful in several areas of computer science, including image processing, robotics, machine learning etc. However, the process is actually very simple and requires a few elementary 2D array operations to be performed. You see, in computer science, images are represented as a 2D array of *pixels*. Each pixel (for sake of simplicity) can be thought of as simply an integer. If you have an image with 20 rows and 30 columns, you will have a total of 20 x 30 = 600 pixels and this image can be represented as a 2D array with 20 rows and 30 columns. Equivalently, when someone says that the size of image is 2048 x 1536 pixels, what they mean is that the matrix of pixels is of the size 2048 x 1536.

The process of convolution is simply that of adding each element of the image to its local neighbors, weighted by a certain thing called a *kernel*. Donâ€™t be scared by that statement. Weâ€™ll explain this to you. As we said, every image can be thought of as just a 2D matrix of pixels. Each pixel has an integer value denoting the color stored at that pixel.

Now lets talk about kernels. A kernel is simply a square matrix with side k, where k is a strictly positive odd integer. Each value in this matrix may be a float number. Applying a kernel to any image means applying the kernel to every pixel of the image. Example: Consider the following 4 x 4 matrix given below. Letâ€™s apply a 3 x 3 kernel to this matrix. We choose a pixel for our demonstration (the one colored red). We align the center of the kernel matrix with the required pixel (the grey 3 x 3 region).

Image:

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34
A41	A42	A43	A44

Kernel:

K11	K12	K13
K21	K22	K23
K31	K32	K33

Applying kernel to pixel 3,3

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34
A41	A42	A43	A44

Applying the filter to pixel (3, 3) updates the value of that pixel according to following equation. Aâ€™ denotes the pixel value in new image.

$$
\begin{aligned}
A'_{3,3} = & \frac{ \Big(\\
& A_{2,2}*K_{1,1} + A_{2,3}*K_{1,2} + A_{2,4}*K_{1,3} + A_{3,2}*K_{2,1} + \\
& A_{3,3}*K_{2,2} + A_{3,4}*K_{2,3} + A_{4,2}*K_{3,1} + A_{4,3}*K_{3,2} + A_{4,4}*K_{3,3} \\
\end{aligned} \Big) \\
}{K_{1,1} + K_{1,2} + K_{1,3} + K_{2,1} + K_{2,2} + K_{2,3} + K_{3,1} + K_{3,2} + K_{1,3}}
\end{aligned}
$$

The above expression may generate a non-integer value which is typecast to int before using it. The above is carried out for all pixels. At the edges, the kernel wraps around and uses element close to opposite edge

Example:

A11	A12	A13	A14
A21	A22	A23	A24
A31	A32	A33	A34
A41	A42	A43	A44

Youâ€™ll receive a image with m x n pixels and a k x k kernel. Youâ€™ve to apply the kernel to the image and print the new image.

Input format
The first line contains 3 integers: m, n, and k.
Next m lines contain n integers each, each giving a row of the image.
Next k lines contain k real values each, each giving a row of the kernel. Note that kernel values are to be interpreted as a float value.

Output format
The output should contain m lines, with n integers on each line. These integers should denote the pixel values of new image.

Caution:

1. Take care of trailing white-spaces.
2. Rest ensured that the summation in the denominator of update equation is never zero.
3. Remember that although the convolution process may generate a floating point number, make sure to cast it into an int before giving your output.
4. In your output, make sure there are no trailing spaces or trailing new lines. Output one row in one line, with a single space between two entries of a row.

Code to manipulate matrices

```
int m, n;
scanf("%d %d", &m, &n);
int num[m][n], i, j;
for(i = 0; i < m; i++)
for(j = 0; j < n; j++)
scanf("%d", &num[i][j]);
```

printf("%d", num[i][j]);

Grading Scheme:
Total marks: **[20 Points]**

There will be partial grading in this question. There are several lines in your output. Printing each line correctly, in the correct order, carries equal weightage. Each visible test case is worth 2 points and each hidden test case is worth 4 points. There are 2 visible and 4 hidden test cases.

Please remember, however, that when you press Submit/Evaluate, you will get a green bar only if all parts of your answer are correct. Thus, if your answer is only partly correct, Prutor will say that you have not passed that test case completely, but when we do autograding afterwards, you will get partial marks.

All Test Cases (Visible + Hidden)

Input	Output
2 4 1 1 2 3 4 4 3 2 1 1	1 2 3 4 4 3 2 1
3 3 3 1 2 3 4 5 6 7 8 9 0 0 0 1 0 0 0 0 0	3 1 2 6 4 5 9 7 8
2 4 3 1 2 3 4 4 3 2 1 -1 -1 -1 -1 1 -1 -1 -1 -1	3 2 2 2 2 2 2 3
3 3 3 1 2 3 4 5 6 7 8 9 1 1 1 1 1 1 1 1 1	5 5 5 5 5 5 5 5 5
5 5 5 36 24 6 24 36 24 16 4 16 24 6 4 1 4 6 24 16 4 16 24 36 24 6 24 36 1 4 6 4 1 4 16 24 16 4 6 24 36 24 6 4 16 24 16 4 1 4 6 4 1	25 19 15 19 25 19 15 12 15 19 15 12 9 12 15 19 15 12 15 19 25 19 15 19 25
4 2 1 0 1	0 1 1 0

1 0 0 1 1 0 -1	0 1 1 0