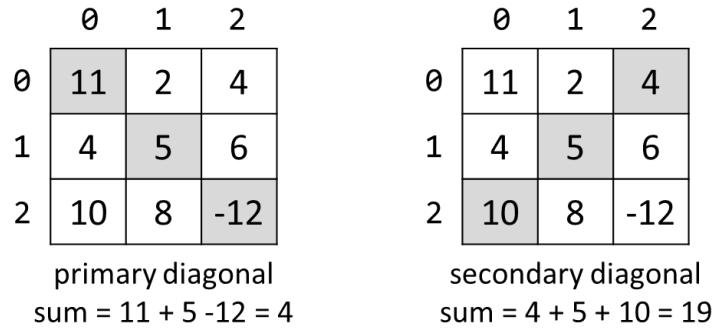
Exercises: Multidimensional Arrays

Problems for exercises and homework for the "CSharp Advanced" course @ Software University.

1. Diagonal Difference

Write a program that finds the **difference between the sums of the square matrix diagonals** (absolute value).



Input

- On the **first line**, you are given the integer **N** – the size of the square matrix
- The next **N lines** holds the values for **every row** – **N** numbers separated by a space

Output

- Print **the absolute** difference between **the sums** of the primary and the secondary diagonal

Examples

Input	Output	Comments
3 11 2 4 4 5 6 10 8 -12	15	**Primary diagonal:** sum = 11 + 5 + (-12) = 4 **Secondary diagonal:** sum = 4 + 5 + 10 = 19 **Difference:** |4 - 19| = 15

2. 2x2 Squares in Matrix

Find the count of **2 x 2 squares of equal chars** in a matrix.

Input

- On the **first line**, you are given the integers **rows** and **cols –** the matrix's dimensions
- Matrix characters come at the next **rows** lines (space separated)

Output

- Print the number of all the squares matrixes you have found

Examples

Input	Output	Comments
3 4 A B B D E B B B I J B B	2	Two 2 x 2 squares of equal cells: A **B B** D E **B B** B I J B B A B B D E B **B B** I J **B B**
2 2 a b c d	0	No 2 x 2 squares of equal cells exist.

3. Maximal Sum

Write a program that reads a rectangular integer matrix of size **N x M** and finds in it the square **3 x 3** that **has maximal sum of its elements**.

Input

- On the first line, you will receive the rows **N** and columns **M**. On the next **N lines** you will receive **each row with its columns**

Output

- Print the **elements** of the 3 x 3 square as a matrix, along with their **sum**

Examples

Input	Matrix	Output
4 5 1 5 5 2 4 2 1 4 14 3 3 7 11 2 8 4 8 12 16 4	1 5 5 2 4 2 **1 4 14** 3 3 **7 11 2** 8 4 **8 12 16** 4	Sum = 75 1 4 14 7 11 2 8 12 16

4. Matrix shuffling

Write a program which reads a string matrix from the console and performs certain operations with its elements. User input is provided in a similar way like in the problems above – first you read the **dimensions** and then the **data**.

Your program should then receive commands in format: "**swap row1 col1 row2c col2**" where row1, row2, col1, col2 are **coordinates** in the matrix. In order for a command to be valid, it should start with the "**swap**" keyword along with **four valid coordinates** (no more, no less). You should **swap the values** at the given coordinates (cell [row1, col1] with cell [row2, col2]) **and print the matrix at each step** (thus you'll be able to check if the operation was performed correctly).

If the **command is not valid** (doesn't contain the keyword "swap", has fewer or more coordinates entered or the given coordinates do not exist), print "**Invalid input!**" and move on to the next command. Your program should finish when the string "**END**" is entered.

Examples

Input	Output
2 3 1 2 3 4 5 6 swap 0 0 1 1 swap 10 9 8 7 swap 0 1 1 0 END	5 2 3 4 1 6 Invalid input! 5 4 3 2 1 6
1 2 Hello World 0 0 0 1 swap 0 0 0 1 swap 0 1 0 0 END	Invalid input! World Hello Hello World

5. Snake Moves

You are walking in the park and you encounter a snake! You are terrified, and you start running zig-zag, so the snake starts following you.

You have a task to visualize the snake's path in a square form. A **snake** is represented by **a string**. The **isle** is a **rectangular matrix of size NxM**. A snake starts climbing the stairs from the **bottom-right corner** and slithers its way up in a **zigzag** – first it moves left until it reaches the left wall, it climbs on the next row and starts moving right, then on the third row, moving left again and so on. The first cell (bottom-right corner) is filled with the first symbol of the snake, the second cell (to the left of the first) is filled with the second symbol, etc. The snake is as long as it takes in order to **fill the stairs completely** – if you reach the end of the string representing the snake, start again at the beginning. After you fill the matrix with the snake's path, you should print it.

Input

- The input data should be read from the console. It consists of exactly three lines
- On the first line, you'll receive the **dimensions** of the stairs in format: **"N M"**, where **N** is the number of **rows**, and **M** is the number of **columns**. They'll be separated by a single space
- On the second line you'll receive the string representing the **snake**

Output

- The output should be printed on the console. It should consist of **N lines**
- Each line should contain a string representing the respective row of the matrix

Constraints

- The **dimensions** N and M of the matrix will be integers in the range [1 … 12]
- The **snake** will be a string with length in the range [1 … 20] and **will not contain any whitespace characters**

Examples

Input	Output	Comments
5 6 SoftUni	SoftUn iSoftU niSoft UniSof tUniSo	

6. Bomb the Basement

You are angry with your neighbor and you wish to get back on him for the constant noise complaints that he files against you. The most valuable things he has are stored in his basement. You have a plan – designing small bombs to bomb it. The basement is in the form of a square. It is full of cells with items.

You will be given the dimensions of the basement. After that you will be given the coordinates of the cells that store the most valuable items – you should bomb them. When a bomb explodes it has an impact and it destroys all of the items in a certain radius, which will be given to you. The items **should be represented by 0**. You can check whether a cell is inside the blast radius using the **Pythagorean Theorem**. The bomb leaves the cells without an item. **You should use the number 1 to represent that**. The items above the exploded area start falling down until they land on another symbol **(meaning an item moves down a row**

as long as there is a -1 below). When the horror ends, print on the console the **resulting basement, each row on a new line**. You should check out the examples.

Input

- The input data should be read from the console. It consists of exactly three lines
- On the first line, you'll receive the **dimensions** of the stairs in format: **"N M"**, where **N** is the number of **rows**, and **M** is the number of **columns**. They'll be separated by a single space
- On the second line, you'll receive the **bomb parameters (target row, target column and radius)**, all separated by a **single space**
- The input data will always be valid and in the format described. There is no need to check it explicitly

Output

- The output should be printed on the console. It should consist of **N lines**
- Each line should contain a string representing the respective row of the final matrix

Constraints

- The **dimensions** N and M of the matrix will be integers in the range [1 … 12]
- The items will represent characters
- The shot's **impact row and column** will always be **valid coordinates** in the matrix – they will be integers in the range [0 … N – 1] and [0 … M – 1] respectively
- The shot's **radius** will be an integer in the range [0 … 4]

Input	Output	Comments
5 6 2 3 2	011111 001110 001110 000100 000100	The matrix has 5 rows and 6 columns. Fill it with zeros: 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 The shot lands on cell (2,3). It has a radius of 2 cells. The impact cell is shaded black and the other cells within the shot radius are shaded grey. 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 Replace all zeros in the blast area with 1: 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 1 1

		0 0 1 1 1 0 0 0 0 1 0 0 The resulting matrix should look like this: 0 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0

7. Knight Game

Chess is the oldest game, but it is still popular these days. For this task we will use only one chess piece – the **Knight**.

The knight moves to the **nearest square but not on the same** row**,** column**, or** diagonal. (This can be thought of as moving two squares horizontally, then one square vertically, or moving one square horizontally then two squares vertically— i.e. in an **"L" pattern**.)

The knight game is played on a board with dimensions **N x N** and a lot of chess knights **0 <= K <= N^2**.

You will receive a board with **K** for knights and '**0**' for empty cells. Your task is to remove a minimum of the knights, so there will be no knights left that can attack another knight.

Input

On the first line, you will receive the **N** size of the board
On the next **N** lines, you will receive strings with **Ks** and **0s**.

Output

Print a single integer with the minimum number of knights that needs to be removed

Constraints

- Size of the board will be 0 < N < 30
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output

5 0K0K0 K000K 00K00 K000K 0K0K0	1
2 KK KK	0
8 0K0KKK00 0K00KKKK 00K0000K KKKKKK0K K0K0000K KK00000K 00K0K000 000K00KK	12

8. *Bombs

You will be given a square matrix of integers, each integer separated by a **single space**, and each row on a new line. Then on the last line of input you will receive indexes - coordinates to several cells separated by a **single space**, in the following format: **row1,column1 row2,column2 row3,column3…**

On those cells there are bombs. You have to proceed **every bomb**, one by one in the order they were given. When a bomb explodes deals damage **equal** to its **own integer value**, to **all** the cells **around** it (in every direction and in all diagonals). One bomb can't explode more than once and after it does, its value becomes **0**. When a cell's value reaches **0 or below**, **it dies**. Dead cells **can't explode**.

You must **print the count of all alive cells** and **their sum**. Afterwards, print the matrix with all of its cells (including the dead ones).

Input

- On the first line, you are given the integer N – the size of the square matrix.
- The next N lines holds the values for every row – N numbers separated by a space.
- On the last line you will receive the coordinates of the cells with the bombs in the format described above.

Output

- On the first line you need to print the count of all alive cells in the format:
  "**`Alive cells: {aliveCells}`**"

- On the second line you need to print the sum of all alive cell in the format: "**`Sum: {sumOfCells}`**"
- In the end print the matrix. The cells must be **separated by a single space**.

Constraints

- The size of the matrix will be between **[0…1000].**
- The bomb coordinates will **always** be in the matrix.
- The bomb's values will always be **greater** than **0**.
- The integers of the matrix will be in range **[1…10000].**

Examples

Input	Output	Comments
4 8 3 2 5 6 4 7 9 9 9 3 6 6 8 1 2 1,2 2,1 2,0	Alive cells: 3 Sum: 12 8 -4 -5 -2 -3 -3 0 2 0 0 -4 -1 -3 -1 -1 2	First the bomb with value 7 will explode and reduce the values of the cells around it. Next the bomb with coordinates 2,1 and value 9 will explode and reduce its neighbour cells. In the end the bomb with coordinates 2,0 and value 9 will explode. After that you have to print the count of the alive cells, which is 3, and their sum is 12. Print the matrix after the explosions.
3 7 8 4 3 1 5 6 4 9 0,2 1,0 2,2	Alive cells: 3 Sum: 8 4 1 0 0 -3 -8 3 -8 0	