## Sum Matrix Elements

Write a program that **reads a matrix** from the console and prints:

* Count of **rows**
* Count of **columns**
* Sum of all **matrix elements**

On the first line, you will get matrix sizes in format **[rows, columns]**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 6  7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0 | 3  6  76 |

### Hints

* On the next **[rows]** lines, you will get elements for each column separated with coma and whitespace.
* Try to use only **foreach** for printing.

## Sum Matrix Columns

Create a program that **reads a matrix** from the console and prints the sum for each column. On the first line, you will get matrix **rows**. On the next **rows** lines, you will get elements for each column separated with a space.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 6  7 1 3 3 2 1  1 3 9 8 5 6  4 6 7 9 1 0 | 12  10  19  20  8  7 |
| 3, 3  1 2 3  4 5 6  7 8 9 | 12  15  18 |

### Hints

* Read matrix sizes.
* On the next lines, read the columns.
* Traverse the matrix and sum all elements in each column.
* Print the sum and continue with the other columns.

## Primary Diagonal

Create a program that finds the **sum of elements from the matrix’s primary diagonal**.



### Input

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

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  11 2 4  4 5 6  10 8 -12 | 4 |
| 3  1 2 3  4 5 6  7 8 9 | 15 |

## Symbol in Matrix

Create a program that reads **N**, a number representing **rows** and **cols** of a **matrix**. On the next **N** lines, you will receive rows of the matrix. Each row consists of ASCII characters. After that, you will receive a symbol. Find the **first occurrence** of that symbol in the matrix and print its position in the format: "({row}, {col})". If there is no such symbol, print an error message "{symbol} does not occur in the matrix".

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  ABC  DEF  X!@  ! | (2, 1) |
| 4  asdd  xczc  qwee  qefw  4 | 4 does not occur in the matrix |

## Square with Maximum Sum

Create a program that **reads a matrix** from the console. Then find the biggest sum of the **2x2 submatrix** and print it to the console.

On the first line, you will get matrix sizes in format **rows,** **columns.**

On the next **rows** lines, you will get elements for each **column,** separated with a comma and a space.

Print the **biggest top-left** square, which you find, and the sum of its elements.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3, 6  7, 1, 3, 3, 2, 1 1, 3, 9, 8, 5, 6 4, 6, 7, 9, 1, 0 | 9 8  7 9  33 |
| 2, 4  10, 11, 12, 13  14, 15, 16, 17 | 12 13  16 17  58 |

### Hints

* Think about **IndexOutOfRangeException().**
* If you find more than one max square, print the top-left one (min row, then min column).

## Jagged-Array Modification

Create a program that **reads a matrix** from the console. On the first line, you will get matrix **rows**. On the next **rows** lines, you will get elements for each **column** separated with **space**. You will be receiving commands in the following format:

* **Add {row} {col} {value}** – **Increase** the number at the given **coordinates** with the **value.**
* **Subtract {row} {col} {value}** – **Decrease** the number at the given **coordinates** by the **value**.

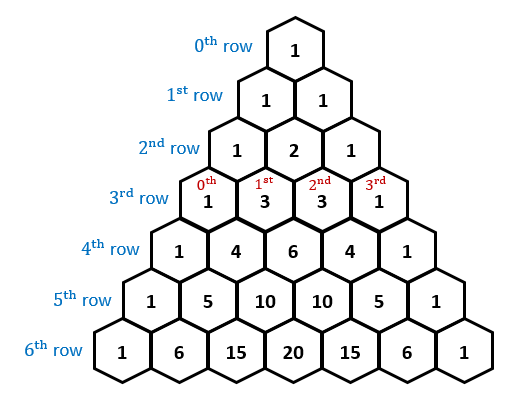
Coordinates might be invalid. In this case, you should print "**Invalid coordinates**". When you receive "**END**" you should print the matrix and stop the program.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  1 2 3  4 5 6 7  8 9 10  Add 0 0 5  Subtract 1 2 2  Subtract 1 4 7  END | Invalid coordinates  6 2 3  4 5 4 7  8 9 10 |
| 4  1 2 3 4  5  8 7 6 5  4 3 2 1  Add 4 4 100  Add 3 3 100  Subtract -1 -1 42  Subtract 0 0 42  END | Invalid coordinates  Invalid coordinates  -41 2 3 4  5  8 7 6 5  4 3 2 101 |

## Pascal Triangle

The **Pascal’s triangle** may be constructed in the following manner: in row 0 (the topmost row), there is a unique nonzero entry 1. Each entry of each subsequent row is constructed by adding the number above and to the left with the number above and to the right, treating blank entries as 0:



If you want more info about Pascal's triangle [here](https://en.wikipedia.org/wiki/Pascal's_triangle).

Write a program to **print the Pascal’s triangle** of given size **n**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 | 1  1 1  1 2 1  1 3 3 1 |
| 13 | 1  1 1  1 2 1  1 3 3 1  1 4 6 4 1  1 5 10 10 5 1  1 6 15 20 15 6 1  1 7 21 35 35 21 7 1  1 8 28 56 70 56 28 8 1  1 9 36 84 126 126 84 36 9 1  1 10 45 120 210 252 210 120 45 10 1  1 11 55 165 330 462 462 330 165 55 11 1  1 12 66 220 495 792 924 792 495 220 66 12 1 |

### Hints

* The input number **n** will be **1 <= n <= 60**.
* Think about the proper **type** for elements in the array.
* Don't be scared to use **more and more arrays**.
* Use a **jagged array**, with triangular form.
* Each row is created by summing two elements from the previous rows.
* You may use