**Exercise: Dynamic Programming Advanced**

This document defines the lab for ["Algorithms – Advanced (Java)" course @ Software University](https://softuni.bg/trainings/2992/algorithms-advanced-with-java-june-2020). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/2513/Dynamic-Programming-Advanced-Exercise).

1. **Abaspa basapa**

You are given 2 strings. Find the left-most **longest ordered common subsequence** in the first string. That means that the elements should be **next to each other** in both strings.

## Input

* On the first line, you will receive the first string
* On the second line, you will receive the second string

## Output

* On a single line, print the **longest subsequence**

## Constraints

* Time limit: **300 ms**. Allowed memory: **64 MB**.

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | |
| abaspa  basapa | bas | |
| **Input** | **Output** |
| ThisIsRandomString  raisstran | is |
| **Input** | **Output** | | |
| abba  baba | ab | | |

# 2. The Mad Gardener

*In the ancient times there were stories about mad gardeners, it was trendy back then. And there was some of us who found some of them somehow special. One among all of them had the most beautiful garden, or at least the story tells so. There were of course any kind of plants, but the work was not at all that easy.*

The garden needs to be cut from time to time. And you have been hired to do some programing models of the garden outlook. Now you will read the **height of the plants as integers** **on a single line**. There of course is something tricky. You have to **remove some of the plants** in such a way that the remaining plants **should increase in height and then begin to decrease at some point**. The garden must however keep its best look so the remaining plants **must have highest possible average height**. And also you have to **remove minimum number of plants**.

## Input

The input will come from the console on a single line.

* On a single line the sequence representing the height of each plant.

## Output

* First print the remaining plants heights on a single line separated by spaces **"{p1 p2 p3 …pn}"**
* On the second line print the average height **formatted to the second symbol of the decimal point**
* On the third line the **count of remaining plants**

## Constraints

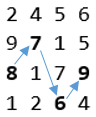
* The number of plants will be in the range **[0…500]**
* The range of the sequences will be **[1…2000]**

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 15 28 7 3 14 4 | 15 28 14 4  15.25  4 |
| 15 28 7 3 14 4 14 16 | 15 28 14 14  17.75  4 |

# Zig-Zag Matrix

You are given a **matrix of positive integer numbers**. A **sig-zag path** in the matrix starts from some cell in the first column, goes to some cell **up** in the second column, then to some cell **down** in the third column, etc. until the last column is reached. Your task is to write a program that finds the zigzag path with the **maximal sum**. Example:



If multiple maximal zigzag paths exist, print the first one which uses the upper-most cell possible at each column (from left to right).

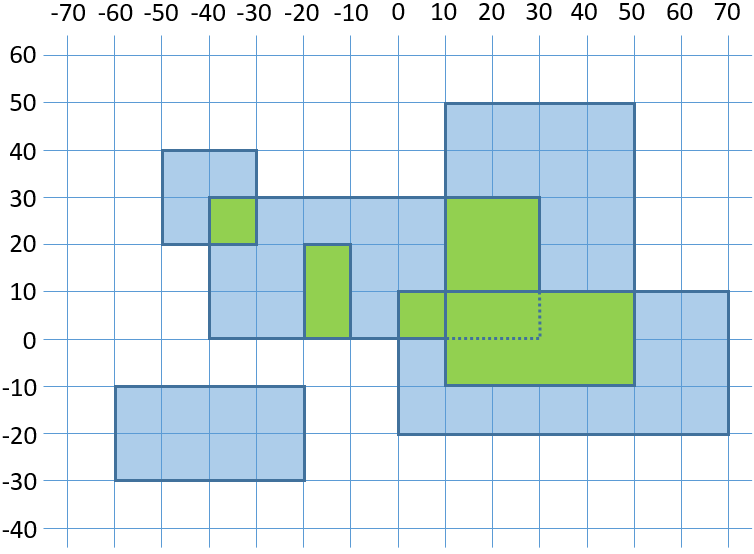
On the first line of input you’ll receive the number of rows N. On the second line you’ll receive the number of columns M. On each of the next N rows you’ll receive M positive integer numbers separated by a comma (',').

Print the maximal sum along with the path followed in format: {maxSum} = {cell1} + {cell2} + … + {cellM}.

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  4  2,4,5,6  9,7,1,5  8,7,7,9  8,2,6,4 | 30 = 8 + 7 + 6 + 9 |
| 17  4  714,52,415,740  102,321,549,697  44,830,171,952  414,58,309,16  956,596,667,526  711,691,776,214  617,919,924,536  102,637,758,360  446,315,243,132  856,313,794,920  732,566,376,314  891,869,999,456  363,869,471,137  650,108,393,24  277,201,124,184  397,13,596,408  73,811,506,100 | 3761 = 891 + 919 + 999 + 952 |
| 5  10  339,575,789,846,979,801,574,337,95,863  612,383,154,963,796,733,748,281,370,854  675,164,992,998,38,958,856,214,567,348  857,709,774,768,270,798,663,440,506,66  458,172,785,558,953,312,854,131,222,250 | 7919 = 857 + 575 + 992 + 963 + 953 + 958 + 854 + 337 + 567 + 863 |

## Rectangle Intersection

You are given N **rectangles** in the plane. The rectangles are parallel to the coordinate axes and each is defined by its coordinates: {minX, maxX, minY, maxY}. Write a program to find the **total area** of all areas that belong to more than one of the initial rectangles. All coordinates are integers in the **range [-1000, 1000]**. Example:



We have 6 rectangles. Their intersection areas are shown in green. The intersection area is 1600.

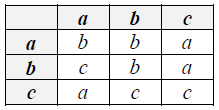
On the first line you’ll receive the **number of rectangles N**. On the next N lines, you’ll receive the coordinates of each rectangle in format {minX} {maxX} {minY} {maxY}. On the only output line, print the total area belonging to more than one rectangle.

**Examples:**

|  |  |
| --- | --- |
| **Input** | **Output** |
| 6  -60 -20 -30 -10  -50 -30 20 40  -40 30 0 30  10 50 -10 50  0 70 -20 10  -20 -10 0 20 | 1600 |
| 3  40 80 -40 0  20 60 -20 30  50 100 -10 20 | 800 |
| 9  -851 88 546 659  990 999 608 998  815 835 -517 734  157 623 994 996  947 956 529 925  561 688 -241 434  -966 530 -825 273  396 780 -705 590  110 202 713 891 | 216777 |

## Symbol Multiplication

We have an **alphabet** of k symbols (a finite number) and a **multiplication table** showing the result of multiplying each two symbols of the alphabet. E.g., the alphabet is {a, b, c} and the multiplication table is:



This shows that a\*a = b, a\*b = b, b\*a = c, etc. As shown in the example, multiplication is **not commutative or associative** – a\*b != b\*a, therefore, the order of multiplication is essential.

We have a string S comprised of characters from the alphabet. The task is to find whether we can obtain the symbol **'a'** by inserting brackets in the string – all symbols in brackets are multiplied. If so, print the string with the brackets inserted. Print "No solution" otherwise. Assume 'a' will always be in the alphabet.

Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| Alphabet = {a,b,c}  Table =  bba  cba  aac  S = abc | ((a\*b)\*c) | ((a\*b)\*c) = (b\*c) = a |
| Alphabet = {a,b,c}  Table =  bba  cba  aac  S = bacacbcabbbcacab | (((b\*a)\*(c\*a))\*(((c\*(b\*c))\*a)\*((b\*((b\*b)\*(c\*a)))\*(c\*(a\*b))))) |  |
| Alphabet = {a,b}  Table =  bb  bb  S = abbbaaba | No solution | No combination of two symbols produces 'a' after multiplication. |