Homework: Arrays, Lists, Stacks, Queues

This document defines the homework assignments from the "Advanced C#" Course @ Software University. Please submit as homework a single **`zip`** / **`rar`** / **`7z`** archive holding the solutions (source code) of all below described problems.

Problem 1. Sort Array of Numbers

Write a program to read an array of numbers from the console, **sort them** and print them back on the console. The numbers should be entered from the console on a single line, separated by a space. Examples:

Input	Output
`6 5 4 10 -3 120 4`	`-3 4 4 5 6 10 120`
`10 9 8`	`8 9 10`

Problem 2. Sort Array of Numbers Using Selection Sort

Write a program to sort an array of numbers and then print them back on the console. The numbers should be entered from the console on a single line, separated by a space. Refer to the examples for problem 1.

Note: Do not use the built-in sorting method, you should **write your own**. Use the **selection sort algorithm**.

***Hint**: To understand the sorting process better you may use a visual aid, e.g. Visualgo.*

Problem 3. Categorize Numbers and Find Min / Max / Average

Write a program that reads N **floating-point numbers** from the console. Your task is to separate them in two sets, one containing only the **round numbers** (e.g. 1, 1.00, etc.) and the other containing the **floating-point numbers with non-zero fraction**. Print both arrays along with their minimum, maximum, sum and average (rounded to two decimal places). Examples:

Input	Output

`1.2 -4 5.00 12211 93.003 4 2.2`	`[1.2, 93.003, 2.2] -> min: 1.2, max: 93.003, sum: 96.403, avg: 32.13`
	
	`[-4, 5, 12211, 4] - > min: -4, max: 12211, sum: 12216, avg: 3054.00`

Problem 4. Sequences of Equal Strings

Write a program that reads an array of strings and finds in it all sequences of equal elements (comparison should be **case-sensitive**). The input strings are given as a single line, separated by a space. Examples:

Input	Output

`hi yes yes yes bye`	`hi`
	`yes yes yes`
	`bye`

SoftUni softUni softuni	SoftUni softUni softuni
1 1 2 2 3 3 4 4 5 5	1 1 2 2 3 3 4 4 5 5
a b b xxx c c c	a b b xxx c c c
hi hi hi hi hi	hi hi hi hi hi
hello	hello

Problem 5. Longest Increasing Sequence

Write a program to find all **increasing** sequences inside an array of integers. The integers are given on a single line, separated by a space. Print the sequences in the order of their appearance in the input array, each at a single line. Separate the sequence elements by a space. Find also the longest increasing sequence and print it at the last line. If several sequences have the same longest length, print the **left-most** of them. Examples:

Input	Output
2 3 4 1 50 **2 3 4 5**	2 3 4 1 50 2 3 4 5 Longest: 2 3 4 5
8 9 9 9 -1 5 2 3	8 9 9 9 -1 5 2 3 Longest: 8 9
1 2 3 4 5 6 7 8 9	1 2 3 4 5 6 7 8 9 Longest: 1 2 3 4 5 6 7 8 9
5 **-1 10 20** 3 4	5 -1 10 20 3 4 Longest: -1 10 20
10 9 8 7 6 5 4 3 2 1	10 9 8 7 6 5

	4 3 2 1 Longest: 10

Problem 6. Subset Sums

Write a program that reads from the console a number N and an array of integers given on a single line. Your task is to find all **subsets** within the array which have a **sum equal to N** and print them on the console (the order of printing is not important). Find only the **unique subsets** by **filtering out repeating numbers first**. In case there aren't any subsets with the desired sum, print **"No matching subsets."** Examples:

Input	Output
11 0 11 1 10 5 6 3 4 7 2	0 + 11 = 11 11 = 11 1 + 10 = 11 0 + 1 + 10 = 11 5 + 6 = 11 0 + 5 + 6 = 11 1 + 6 + 4 = 11 0 + 1 + 6 + 4 = 11 1 + 3 + 7 = 11 0 + 1 + 3 + 7 = 11 4 + 7 = 11 0 + 7 + 4 = 11 1 + 5 + 3 + 2 = 11 0 + 1 + 5 + 3 + 2 = 11 6 + 3 + 2 = 11 0 + 6 + 3 + 2 = 11 5 + 4 + 2 = 11 0 + 5 + 4 + 2 = 11
0 1 2 3 4 5	No matching subsets.
-2 -5 4 92 0 928 1 -1 4	-5 + 4 + -1 = -2 -5 + 4 + 0 + -1 = -2

Hint: Search how to generate subsets of elements with a bitwise mask.

Problem 7. * Sorted Subset Sums

Modify the program you wrote for the previous problem to print the results in the following way: each line should contain the **operands** (numbers that form the desired sum) in **ascending order**; the lines **containing fewer operands** should be printed **before** those with more operands; when two lines have the same number of operands, the one

containing the **smallest operand** should be printed first. If two or more lines contain the same number of operands and have the same smallest operand, the order of printing is not important. Example:

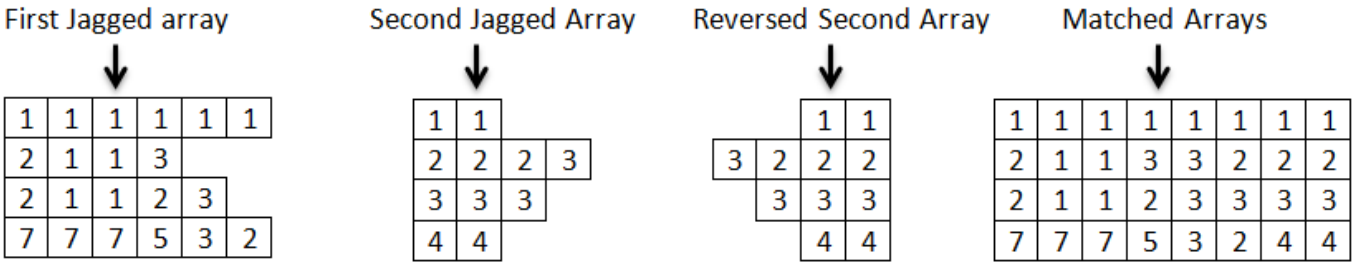
Input	Output
11 0 11 1 10 5 6 3 4 7 2	11 = 11 -> *only one operand* 0 + 11 = 11 -> *two operands, smallest is 0* 1 + 10 = 11 -> *two operands, smallest is 1* 4 + 7 = 11 5 + 6 = 11 0 + 5 + 6 = 11 -> *three operands, smallest is 0* 0 + 4 + 7 = 11 -> *this line can be switched with the one above or with the one below (they all have three operands, smallest is 0)* 0 + 1 + 10 = 11 1 + 4 + 6 = 11 1 + 3 + 7 = 11 2 + 4 + 5 = 11 2 + 3 + 6 = 11 0 + 2 + 4 + 5 = 11 0 + 2 + 3 + 6 = 11 0 + 1 + 4 + 6 = 11 0 + 1 + 3 + 7 = 11 1 + 2 + 3 + 5 = 11 0 + 1 + 2 + 3 + 5 = 11
0 1 2 3 4 5	No matching subsets.
-2 -5 4 92 0 928 1 -1 4	-5 + -1 + 4 = -2 -5 + -1 + 0 + 4 = -2

Ideas to help you out: You'll probably want to store all subset sums in a collection. You'll need to check if the collection has elements after you've checked all combinations for valid subsets. If the collection contains elements, you need to sort it by the criteria provided above. LINQ extension methods and lambda expressions can do this in just a few lines of code!

Problem 8. * Lego Blocks

This problem is from the Java Basics Exam (8 February 2015). You may check your solution here.

You are given two **jagged arrays**. Each array represents a **Lego block** containing integers. Your task is first to **reverse** the second jagged array and then check if it would **fit perfectly** in the first jagged array.



The picture above shows exactly what fitting arrays mean. If the arrays fit perfectly you should **print out** the newly made rectangular matrix. If the arrays do not match (they do not form a rectangular matrix) you should print out the **number of cells** in the first array and in the second array combined. The examples below should help you understand the assignment better.

Input

The first line of the input comes as an **integer number n** saying how many rows are there in both arrays. Then you have **2 * n** lines of numbers separated by whitespace(s). The first **n** lines are the rows of the first jagged array; the next **n** lines are the rows of the second jagged array. There might be leading and/or trailing whitespace(s).

Output

You should print out the combined matrix in the format:

[elem, elem, …, elem]
[elem, elem, …, elem]
[elem, elem, …, elem]

If the two arrays do not fit you should print out : **`The total number of cells is: `*`count`***

Constraints

- The number n will be in the range [2 … 10].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
2 1 1 1 1 1 1 2 1 1 3 1 1 2 2 2 3	[1, 1, 1, 1, 1, 1, 1, 1] [2, 1, 1, 3, 3, 2, 2, 2]
2 1 1 1 1 1 1 1 1 1 1 1 1 1 1	The total number of cells is: 14

Problem 9. * Stuck Numbers

This problem is from the Java Basics Exam (1 June 2014). You may check your solution here.

You are given **n numbers**. Write a program to find among these numbers all sets of 4 numbers {**a**, **b**, **c**, **d**}, such that **a**|**b** == **c**|**d**, where **a** ≠ **b** ≠ **c** ≠ **d**. Assume that "**a**|**b**" means to append the number **b** after **a**. We call these numbers {**a**, **b**, **c**, **d**} **stuck numbers**: if we append **a** and **b**, we get the same result as if we appended **c** and **d**. Note that the numbers **a**, **b**, **c** and **d** should be distinct (**a** ≠ **b** ≠ **c** ≠ **d**).

Input

The input comes from the console. The first line holds the **count n**. The next line holds **n integer numbers**, separated by a space. The input numbers will be **distinct** (no duplicates are allowed).

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print at the console all **stuck numbers** {**a**, **b**, **c**, **d**} found in the input sequence in format "**a**|**b**==**c**|**d**" (without any spaces), each at a separate line. The **order** of the output lines **is not important**. Print "**No**" in case no stuck numbers exist among the input sequence of numbers.

Constraints

- The **count n** will be an integer number in the range [1 … 50].
- The input **numbers** will be **distinct** integers in the range [0 … 9999].
- Time limit: 0.5 sec. Memory limit: 16 MB.

Examples

Input	Output
5 2 51 1 75 25	2|51==25|1 25|1==2|51

Input	Output
7 2 22 23 32 322 222 5	2|322==23|22 23|22==2|322 32|22==322|2 32|222==322|22 322|2==32|22 322|22==32|222

Input	Output
3 5 1 20	No

Problem 10.* Pythagorean Numbers

This problem is from the Java Basics Exam (26 May 2014). You may check your solution here.

George likes math. Recently he discovered an interesting property of the Pythagorean Theorem: there are infinite number of triplets of integers **a**, **b** and **c** (a ≤ b), such that $\mathbf{a^2 + b^2 = c^2}$. Write a program to help George find all such triplets (called Pythagorean numbers) among a set of integer numbers.

Input

The input data should be read from the console. At the first line, we have a number **n** – the count of the input numbers. At the next **n** lines we have **n different integers**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print at the console all Pythagorean equations $\mathbf{a^2 + b^2 = c^2}$ (a ≤ b), which can be formed by the input numbers. Each equation should be printed in the following format: "**a*a + b*b = c*c**". The order of the equations is not important. Beware of **spaces**: put spaces around the "**+**" and "**=**". In case of no Pythagorean numbers found, print "**No**".

Constraints

- All input numbers will be **unique** integers in the range [0 … 999].
- The **count** of the input numbers will be in the range [1 ... 100].
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output
8 41 5 9 12 4 13 40 3	5*5 + 12*12 = 13*13 9*9 + 40*40 = 41*41 3*3 + 4*4 = 5*5

Input	Output
5 3 12 5 0 4	3*3 + 4*4 = 5*5 0*0 + 3*3 = 3*3 0*0 + 12*12 = 12*12 0*0 + 5*5 = 5*5 0*0 + 0*0 = 0*0 0*0 + 4*4 = 4*4

Input	Output
3 10 20 30	No