More Exercises: Arrays and Methods

Problems for exercises and homework for the "Programming Fundamentals" course @ SoftUni.

Problem 1. Array Statistics

Write a program which receives array of integers (space-separated) and prints the minimum and maximum number, the sum of the elements and the average value.

Examples

Input	Output
234561	Min = 1
	Max = 6
	Sum = 21
	Average = 3.5

Input	Output
-1 200 124123 -400 -124214	Min = -124214
	Max = 124123
	Sum = -292
	Average = -58.4

Problem 2. Manipulate Array

You will receive an array of strings and you have to execute some command upon it. You can receive three types of commands:

- Reverse reverse the array
- Distinct delete all non-unique elements from the array
- Replace {index} {string} replace the element at the given index with the string, which will be given to you

Input

- On the first line, you will receive the string array
- On the **second line**, you will receive **n** the number of **lines**, which will **follow**
- On the next **n lines** you will receive **commands**

Output

At the end print the array in the following format:

{1st element}, {2nd element}, {3rd element} ... {nth element}

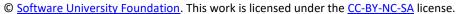
Constraints

- For **separator** will be used only **single whitespace**
- n will be integer in the interval [1...100]

Examples

Input	Output
one one one two three four five	five, four, Hello, two, one
3	
Distinct	
Reverse	
Replace 2 Hello	





















a, Bravo, Charlie, Charlie, Foxtrot

Problem 3. Safe Manipulation

Now we need to make our program safer and more user-friendly. Make the program print "Invalid input!" if we try to replace an element at a non-existent index or an invalid command is written on the console. Also make the program work until the command "END" is given as an input.

Input

- On the first line, you will receive the string array
- On the next lines until you receive "END" you will receive commands

Output

At the end, print the array in the following format:

{1st element}, {2nd element}, {3rd element} ... {nth element}

Constraints

- Only a single whitespace will be used for the separator.
- n will be an integer in the interval [1...100]

Examples

Input	Output
one one two three four five Distinct Reverse Replace 7 Hello Replace -5 Hello Replace 0 Hello END	Invalid input! Invalid input! Hello, four, three, two, one
Input	Output
Alpha Bravo Charlie Delta Echo Foxtrot Distinct Reverse Replace 0 Charlie Reverse Replace 1 Charlie Distinct Replace 4 Charlie END	Invalid input! Invalid input! Alpha, Charlie, Delta, Echo



















Problem 4. Grab and Go

Write a program, which receives an array of integers and an integer as input. Find the last occurrence of the integer in the given array and **print** the **sum** of all elements **before** the **number**.

Example: if we receive the array 10 20 30 40 20 30 40 and we receive on the next line the integer 20 we have to print the sum the elements 10 20 30 40, which is 100.

If no such number exists in the array – print "No occurrences were found!".

Input

- On the first line, you will receive the integer array
- On the next line, you will receive the number, which you have to search

Output

If such number **exists** in the array – just print the **sum**.

Otherwise, print "No occurrences were found!"

Constraints

- Only a **single whitespace** will be used for the **separator**.
- The number will be integer in the interval [-2147483648...2147483647]

Examples

Input	Output
1 3 5 7 12 2 3 5 12 3	30

Input	Output
1 2 3 4 5 6 7 8 9 10 20	No occurrences were found!

Problem 5. Pizza Ingredients

You manage your own pizza restaurant and you are in charge of the orders. Your pizza is made only from ingredients, which have a specific length.

On the first line, you will receive an array of strings with the possible ingredients. On the next line, you will receive an integer, which represents the length of the strings, which we will used in the recipe.

Your recipe should not use more than 10 ingredients. If you collect 10 ingredients stop the program and print the result.

Input

- On the first line, you will receive the ingredients
- On the second line, you will receive the searched length.

Output

Every time you find a **matching ingredient** print:





















Adding {name of the ingredient}.

Print the **answer** in the following format:

Made pizza with total of {count of the ingredients} ingredients.

The ingredients are: {1st ingredient}, {2nd ingredient}, ... {nth ingredient}.

Constraints

- Only a **single whitespace** will be used for the **separator**.
- The array will be with at most 100 elements long.
- **Each ingredient** will be at most 50 characters long.
- The maximum length will be in the interval [1...50]
- You will receive at least one valid ingredient

Examples

Input	Output
cheese flour tomato bread olives salami pepperoni 6	Adding cheese. Adding tomato. Adding olives. Adding salami. Made pizza with total of 4 ingredients. The ingredients are: cheese, tomato, olives, salami.

Input	Output
cheese flour tomato bread olives salami pepperoni	Adding pepperoni. Made pizza with total of 1 ingredients. The ingredients are: pepperoni.

Problem 6. Heists

You are the main accountant for a group of bandits, whose main line of work is robbing banks and stores. Your job is to calculate the score from the heist, calculating the price of the loot and taking the expenses into account.

On the first line, you will receive an array with two elements. The first element represents the price of the jewels and the second – the price of the gold.

On each of the next lines, you will receive input in the format "{loot} {heist expenses}" until you receive the command "Jail Time". The loot will be a string containing random characters. The jewels will be represented with the character "%" and the gold – with the character "\$". If you find either from the symbols it means you have found one of the goodies.

Upon receiving "Jail Time" you have to calculate the total earnings and the total expenses from the heists. If the total earnings are more or equal to the total expenses print: "Heists will continue. Total earnings: {money earned}.". Otherwise print: "Have to find another job. Lost: {money lost}.".

Input

- On the **first line**, you will receive an array of integers with two elements.
 - The first element is the price of the jewels.
 - The second element is the price of the gold.



















- Each of the next lines will contain information in the following format "{loot} {heist expenses}"
 - The **loot** will be a string of random characters.
 - The **heist expenses** will be an **integer** number.
- The last line of the input will always be "Jail Time" signaling the end of the input.

Output

The output should consist of only one line:

- If the total earnings are more or equal to the expenses print:
 - "Heists will continue. Total earnings: {money earned}."
- Alternatively, if the expenses are more than the total earnings print:
 - "Have to find another job. Lost: {money lost}."

Constraints

- Only a **single whitespace** will be used for the **separator**.
- The array will have at most 100 elements.
- You will receive at most 20 heists.
- You will receive at least one valid loot item.
- The heist expenses will be in the interval [1...2147483647].
- The **gold** and **jewel** prices will be **integers** in the interval [1...2147483647].

Examples

Input	Output	Comments
10 20 ASDA% 50 DaS@!%\$\$ 10 \$\$ 10 Jail Time	Heists will continue. Total earnings: 30.	We have price of the jewels of 10 and price of the gold of 20. In the first heist, we found one jewel (total earnings of 10), but the expenses are 50. 2 nd heist -> 2 gold and 1 jewel -> total earnings = 50 + 10 (from the previous heist) and expenses of 10 + 50 (from previous heist) 3 rd heist -> 2 gold -> total earnings = 100; total expenses: 10 + 60 = 70. Total: 100 (total earnings) - 70 (total expenses) = 30

Input	Output
2000 10000 ASDAs 500000 %ASD\$ 1000000 \$S\$&*_ASD 1000 AF#^&*LP 20000 \$ 100000000 Jail Time	Have to find another job. Lost: 101479000.

Hints

- In C#, you can treat strings like arrays of chars and loop through every single element
- In Java, you can take the length of the string, using **String.length()** and access the characters, using String.charAt(index)



















Problem 7. Inventory Matcher

You will be given three arrays on different lines. The first one will contain strings, which will represent the name of products. Second one will contain longs and will represent the quantities of the products. The third one will contain double and represents the price of the product.

After which, you will be given names of products on new lines, until you receive the command "done". For each given product name print:

{name of the product} costs: {price}; Available quantity: {quantity}

The names, prices and quantities of the products are in the same indices in the 3 arrays.

Input

- On the **first line**, you will receive an array of **strings**, which represent the **names** of the products.
- On the **second line**, you will receive an array of **longs**, which represent the **quantities** of the products.
- On the **third line**, you will receive an array of **decimals**, which represent the **prices** of the products.

Constraints

- The **three** arrays will **always** have the **same** length.
- You will always receive existing products.

Examples

Input	Output
Bread Juice Fruits Lemons 10 50 20 30 2.34 1.23 3.42 1.50 Bread Juice done	Bread costs: 2.34; Available quantity: 10 Juice costs: 1.23; Available quantity: 50
Oranges Apples Nuts 1500 5000000 200000000 2.3412 1.23 3.4250 Nuts done	Nuts costs: 3.4250; Available quantity: 2000000000

Hints

- In C#, you can find the index of an element in an array with Array. IndexOf(array, element)
- In Java, the simplest way to find the index of the element (without external libraries) will be to loop through the array

Problem 8. * Upgraded Matcher

For this task, you can use your solution from Inventory Matcher. You will again receive 3 arrays – one with strings, one with longs and one with decimals. Again, the price and quantity correspond to a name, which is located on same index as the name.

This time only the arrays containing the names and the array containing the prices will have the same length. If in the quantities array there is no index, which corresponds to the name, you should assume the quantity is 0.

















On top of that the products, which you receive after the arrays will contain **not only** a string for the **name**, but also a long, which is the quantity that must be ordered.

If you have enough quantity, calculate the total price by multiplying the ordered quantity times the price and print it in the following format:

{product name} x {quantity ordered} costs {total price of the order}

Format the price to the 2nd decimal place. Do not forget to decrease the quantity of the product.

If you do not have enough quantities print:

We do not have enough {product name}

Input

- On the **first line**, you will receive array of **strings**, which represent the **names** of the products.
- On the **second line**, you will receive array of **longs**, which represent the **quantities** of the products.
- On the **third line**, you will receive array of **decimals**, which represent the **prices** of the products.

Constraints

- The **name** and **price arrays** will **always** have the **same** length.
- You will always receive existing products

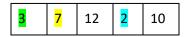
Examples

Input	Output
Bread Juice Fruits Lemons Beer 10 50 20 30 2.34 1.23 3.42 1.50 3.00 Bread 10 Juice 5 Beer 20 done	Bread x 10 costs 23.40 Juice x 5 costs 6.15 We do not have enough Beer
Tomatoes Onions Lemons 10000 2000 5.40 3.20 2.20 Tomatoes 5000 Tomatoes 5000 Tomatoes 1 done	Tomatoes x 5000 costs 27000.00 Tomatoes x 5000 costs 27000.00 We do not have enough Tomatoes

Problem 9. * Jump Around

You will receive an integer array from the console. You start from the beginning of the array and try to move right by a step, equal to the value at position 0. If that is possible you should collect the value from the index on which you landed, and try to move to the **right** by **its value**, if that is **not** possible – try to move to the **left**. If that is also **not** possible **stop** the program and print the **sum** of the collected **values**. Example:

Example: We have the array [3 7 12 2 10]. We start from 3 and move 3 indices to 2. We have to move 2 indices, but we can't move to the right, so we move to the left to 7. From there we cannot move anywhere and we stop the program and we print the sum of the collected cells: 3 + 2 + 7 = 12

















Input

The input consists of **single** line, which will be an **array** of **integers**.

Constraints

- The array will have at most **50** elements
- The elements in the array will be in the interval [1...50]

Examples

Input						Output	
10	50	7	30	8	5		10

Input	Output
2 3 5 7 5 4 8 4	18

















