Exercise: Arrays

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

You can check your solutions in Judge.

1. Train

You will be given a count of wagons in a train **n**. On the next **n** lines, you will receive how many people will get on that wagon. In the end, print the whole train and the sum of the people on the train.

Examples

Input	Output
3 13 24 8	13 24 8 45
6 3 52 71 13 65 4	3 52 71 13 65 4 208
1 100	100 100

2. Common Elements

Write a program that prints common elements in two arrays. You have to compare the elements of the second array to the elements of the first.

Examples

Input	Output
Hey hello 2 4 10 hey 4 hello	4 hello
S of t un i of i 10 un	of i un
i love to code code i love to	code i love to

3. Zig-Zag Arrays

Write a program that creates 2 arrays. You will be given an integer **n**. On the next **n** lines, you get 2 integers. Form 2 arrays as shown below.

Examples

Input	Output
4 1 5 9 10 31 81 41 20	1 10 31 20 5 9 81 41
2 80 23 31 19	80 19 23 31

4. Array Rotation

Write a program that receives an array and the number of rotations you have to perform (the first element goes at the end). Print the resulting array.

Examples

Input	Output
51 47 32 61 21 2	32 61 21 51 47
32 21 61 1 4	32 21 61 1
2 4 15 31 5	4 15 31 2

5. Top Integers

Write a program to find all the top integers in an array. A top integer is an integer that is **bigger** than all the elements to its right.

Examples

Input	Output
1 4 3 2	4 3 2
14 4 3 19 15 17	24 9 17
27 9 42 2 13 45 48	48

6. Equal Sums

Write a program that determines if an **element exists in the array** such that the **sum of the elements on its left** is **equal** to the **sum of the elements on its right**. If there are **no elements to the left/right**, their **sum is considered to be 0**. Print the **index** that satisfies the required condition or "**no**" if there is no such index.

Examples

Input	Output	Comments
1 2 3 3	2	At a[2] -> left sum = 3, right sum = 3 a[0] + a[1] = a[3]
1 2	no	At a[0] -> left sum = 0, right sum = 2 At a[1] -> left sum = 1, right sum = 0 No such index exists
1	0	At a[0] -> left sum = 0, right sum = 0
1 2 3	no	No such index exists
10 5 5 99 3 4 2 5 1 1 4	3	At a[3] -> left sum = 20, right sum = 20 a[0] + a[1] + a[2] = a[4] + a[5] + a[6] + a[7] + a[8] + a[9] + a[10]

7. Max Sequence of Equal Elements

Write a program that finds the **longest sequence of equal elements** in an array of integers. If several longest sequences exist, print the leftmost one.

Examples

Input	Output
2 1 1 2 3 3 **2 2 2** 1	2 2 2
1 1 1 2 3 1 3 3	1 1 1
4 4 4 4	4 4 4 4
0 **1 1** 5 2 2 6 3 3	1 1

8. Magic Sum

Write a program that prints all unique pairs in an array of integers whose sum is equal to a given number.

Examples

Input	Output
1 7 6 2 19 23 8	1 7 6 2
14 20 60 13 7 19 8 27	14 13 20 7 19 8

9. Array Modifier

You are given **an array with integers**. Write a program to **modify the elements** after **receiving the following commands**:

- **"swap {index1} {index2}"** takes **two elements** and **swap their places**.
- **"multiply {index1} {index2}"** takes the **element at the 1st index** and **multiplies it with the element at 2nd index**. **Save the product at the 1st index.**
- **"decrease" decreases all elements** in the array **with 1**.

Input

On the **first input line,** you will be given **the initial array values** separated by a single space.

On the **next lines,** you will receive commands **until** you receive the **command "end"**. The **commands are** as follows:

- `"swap {index1} {index2}"`
- `"multiply {index1} {index2}"`
- `"decrease"`

Output

The output should be printed on the console and consist of **elements of the modified array** – **separated by a comma and a single space** `", "`.

Constraints

- **Elements of the array** will be **integer numbers** in the range **[-2^{31}...2^{31}].**
- **The count of the array elements** will be in the range **[2...100].**
- **Indexes will always be in the range of the array.**

Examples

Input	Output	Comments
23 -2 321 87 42 90 -123 swap 1 3 swap 3 6 swap 1 0 multiply 1 2 multiply 2 1 decrease end	86, 7382, 2369942, -124, 41, 89, -3	23 -2 321 87 42 90 -123 – initial values swap 1(-2) and 3(87) ▼ 23 87 321 -2 42 90 -123 swap 3(-2) and 6(-123) ▼ 23 87 321 -123 42 90 -2 swap 1(87) and 0(23) ▼ 87 23 321 -123 42 90 -2 multiply 1(23) 2(321) = 7383 ▼ 87 7383 321 -123 42 290 -2 multiply 2(321) 1(7383) = 2369943 ▼ 87 7383 2369943 -123 42 90 -2 decrease – all - 1 ▼ 86 7383 2369942 -124 41 89 -3
1 2 3 4 swap 0 1 swap 1 2 swap 2 3 multiply 1 2 decrease end	1, 11, 3, 0	

10. Treasure Hunt

The pirates must safely carry a treasure chest back to the ship, looting along the way.

Create a program that **manages** the **state** of the **treasure chest** along the way. On the **first line,** you will receive the **initial loot** of the treasure chest, a **string** of **items** separated by a **"|"**.

"{$loot_1$}|{$loot_2$}|{$loot_3$} … {$loot_n$}"

The following lines represent commands **until "Yohoho!"** which ends the treasure hunt:

- **"Loot {item$_1$} {item$_2$}…{item$_n$}"**:
  - Pick up treasure loot along the way. Insert the items at the **beginning** of the chest.
  - If an item is **already** contained, **don't** insert it.
- **"Drop {index}"**:
  - **Remove** the loot at the given **position** and **add** it to the **end** of the treasure chest.
  - If the index is **invalid,** skip the command.
- **"Steal {count}"**:
  - Someone steals the **last count** loot items. If there are **fewer items** than the given count, **remove as many** as there are.
  - Print the stolen items separated by **", "**:
    "{item$_1$}, {item$_2$}, {item$_3$} … {item$_n$}"

In the end, output the **average treasure gain,** which is the **sum** of all treasure items **length** divided by the **count** of all items inside the chest **formatted** to the **second decimal** point:

"Average treasure gain: {averageGain} pirate credits."

If the chest is **empty,** print the following message:

"Failed treasure hunt."

Input

- On the **1st line,** you will receive the **initial treasure chest** (**loot** separated by **"|"**).
- On the following **lines**, you will receive commands until **"Yohoho!"**.

Output

- Print the output in the **format described above**.

Constraints

- The **loot items** will be strings containing any ASCII code.
- The **indexes** will be integers in the range [**-200**…**200**].
- The **count** will be an integer in the range [**1**….**100**].

Examples

Input	Output
Gold|Silver|Bronze|Medallion|Cup Loot Wood Gold Coins Loot Silver Pistol Drop 3 Steal 3 Yohoho!	Medallion, Cup, Gold Average treasure gain: 5.40 pirate credits.
Comments	

The first command, **"Loot Wood Gold Coins"** adds **Wood** and **Coins** to the chest but **omits** Gold since it is already contained. The chest now has the following items:

Coins Wood Gold Silver Bronze Medallion Cup

The **second** command adds **only Pistol** to the chest

The **third** command, **"Drop 3"** removes the **Gold** from the chest but immediately adds it at the **end**:

Pistol Coins Wood Silver Bronze Medallion Cup Gold

The **fourth** command, **"Steal 3"** removes the **last 3** items **Medallion**, **Cup**, **Gold,** from the chest and prints them.

In the end calculate the average treasure gain which is the sum of all items length Pistol(**6**) + Coins(**5**) + Wood(**4**) + Silver(**6**) + Bronze(**6**) = **27** and **divide** it by the count 27 / 5 = **5.4** and format it to the **second decimal** point.

Input	Output
Diamonds|Silver|Shotgun|Gold Loot Silver Medals Coal Drop -1 Drop 1 Steal 6 Yohoho!	Coal, Diamonds, Silver, Shotgun, Gold, Medals Failed treasure hunt.