Exercise: Arrays

Problems for exercises and homework for the ["Programming Fundamentals" course @ SoftUni](https://softuni.bg).

You can check your solutions in [Judge](https://judge.softuni.bg).

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. The Lift

Write a program that **finds a place for the tourist on a lift.**

Every wagon should have **a maximum of 4 people on it**. If a wagon is full, you should direct the people to **the next one with space** available.

Input

- **On the first line,** you will receive **how many people** are waiting to get **on the lift.**
- **On the second line**, you will receive the **current state of the lift separated by a single space: " "**.

Output

When there is no more available space left on the lift, or there are **no more people in the queue**, you should print on the console the final state of the lift's wagons separated by a single space **" "** and one of the following messages:

- If there are no more people and the lift has empty spots, you should print:

  ```
  "The lift has empty spots!
  {wagons separated by ' '}"
  ```

- If there are still people in the queue and no more available space, you should print:

  ```
  "There isn't enough space! {people} people in a queue!
  {wagons separated by ' '}"
  ```

- If the lift is full and there are no more people in the queue, you should print only the wagons separated by a single space **" ".**

Examples

<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>15
0 0 0 0</td><td>The lift has empty spots!
4 4 4 3</td></tr>
<tr><th colspan="2">Comment</th></tr>
<tr><td colspan="2">First state - 4 0 0 0 -> 11 people left
Second state – 4 4 0 0 -> 7 people left
Third state – 4 4 4 0 -> 3 people left</td></tr>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>20
0 2 0</td><td>There isn't enough space! 10 people in a queue!
4 4 4</td></tr>
<tr><th colspan="2">Comment</th></tr>
<tr><td colspan="2">First state - 4 2 0 -> 16 people left
Second state – 4 4 0 -> 14 people left
Third state – 4 4 4 -> 10 people left, but they're no more wagons.</td></tr>
</table>