

# JS Fundamentals Mid Exam Preparation

## Problem 1. SoftUni Reception

Link: <https://judge.softuni.org/Contests/Practice/Index/2474#0>

*Every day, thousands of students pass by the reception at SoftUni with different questions to ask. The employees have to help everyone by providing all the information and answering all of the questions.*

**Three employees** are working on the reception all day. Each of them can handle a **different number of students per hour**. Your task is to **calculate how much time** it will take to **answer all the questions** of a given number of students.

First, you will receive 3 lines with integers, representing the number of students that each **employee can help per hour**. On the following line, you will receive **students count as a single integer**.

Every fourth hour, all employees have a break, so they don't work for an hour. It is the only break for the employees, because they don't need rest, nor have a personal life. Calculate the time needed to answer all the student's questions and print it in the following format: **"Time needed: {time}h."**

### Input / Constraints

- On the first three lines - **each employee efficiency** - integer in the range [1 - 100]
- On the fourth line - **students count** – integer in the range [0 – 10000]
- Input will always be valid and in the range specified

### Output

- Print a single line: **"Time needed: {time}h."**
- Allowed working **time / memory**: 100ms / 16MB

### Examples

Input	Output	Comment
['5', '6', '4', '20']	Time needed: 2h.	All employees can answer 15 students per hour. After the first hour, there are 5 students left to be answered. All students will be answered in the second hour.
['1', '2', '3', '45']	Time needed: 10h.	All employees can answer 6 students per hour. In the first 3 hours, they have answered $6 * 3 = 18$ students. <b>Then they have a break for an hour.</b> After the <b>next 3 hours</b> , there are $18 + 6 * 3 = 36$ answered students. After the break for an hour, there are only 9 students to answer. So in the <b>10th hour</b> , all of the student's questions would be answered.

['3','2','5','40']	Time needed: 5h.	
--------------------	------------------	--

## Problem 2. Array Modifier

Link: <https://judge.softuni.org/Contests/Practice/Index/2474#1>

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 element at the 1<sup>st</sup> index and multiply it with the element at 2<sup>nd</sup> index. Save the product at the 1<sup>st</sup> 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 follow:

- "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  $[-2^{31}...2^{31}]$
- Count of the array elements will be in the range  $[2...100]$
- Indexes will be always in the range of the array

## Examples

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

## Problem 3. Heart Delivery

Link: <https://judge.softuni.org/Contests/Practice/Index/2031#2>

*Valentine's day is coming, and Cupid has minimal time to spread some love across the neighborhood. Help him with his mission!*

You will receive a **string** with **even integers**, separated by a "@" - this is our neighborhood. After that, a series of **Jump** commands will follow until you receive "**Love!**". Every house in the neighborhood needs a certain number of **hearts** delivered by Cupid so it can celebrate Valentine's day. The integers in the neighborhood indicate those needed hearts.

Cupid starts at the position of the **first house** (index 0) and must jump by a **given length**. The jump commands will be in this format: "**Jump {length}**".

Every time he jumps from one house to another, the needed hearts for the visited house are **decreased by 2**:

- If the needed hearts for a certain house become **equal to 0**, print on the console "**Place {house\_index} has Valentine's day.**"
- If **Cupid** jumps to a house where the needed hearts are **already 0**, print on the console "**Place {house\_index} already had Valentine's day.**"
- Keep in mind that **Cupid** can have a **larger jump length** than the **size of the neighborhood**, and if he does jump **outside** of it, he should **start** from the **first house** again (index 0)

For example, we are given this neighborhood: 6@6@6. Cupid is at the start and jumps with a length of 2. He will end up at index 2 and decrease the needed hearts by 2: [6, 6, 4]. Next, he jumps again with a length of 2 and goes outside the neighborhood, so he goes back to the first house (index 0) and again decreases the needed hearts there: [4, 6, 4].

## Input

- On the first line, you will receive a **string** with **even integers** separated by "@" – the neighborhood and the number of hearts for each house.
- On the next lines, until "Love!" is received, you will be getting jump commands in this format: "Jump {length}".

## Output

In the end, print **Cupid's last position** and whether his mission was successful or not:

- "Cupid's last position was {last\_position\_index}."
- If **each house** has had Valentine's day, print:
  - "Mission was successful."
- If **not**, print the **count** of all houses that **didn't** celebrate Valentine's Day:
  - "Cupid has failed {houseCount} places."

## Constraints

- The **neighborhood's** size will be in the range [1...20]
- Each **house** will need an **even number** of hearts in the range [2 ... 10]
- Each **jump length** will be an integer in the range [1 ... 20]

## Examples

Input	Output	Comments
[("10@10@10@2", "Jump 1", "Jump 2", "Love!")]	Place 3 has Valentine's day. Cupid's last position was 3. Cupid has failed 3 places.	Jump 1 ->> [10, 8, 10, 2] Jump 2 ->> [10, 8, 10, 0] so we print "Place 3 has Valentine's day." The following command is "Love!" so we print Cupid's last position and the outcome of his mission.
(["2@4@2", "Jump 2", "Jump 2", "Jump 8", "Jump 3", "Jump 1", "Love!")]	Place 2 has Valentine's day. Place 0 has Valentine's day. Place 0 already had Valentine's day. Place 0 already had Valentine's day. Cupid's last position was 1. Cupid has failed 1 places.	
(["4@2@4@2", "Jump 1", "Jump 2", "Jump 1", "Jump 2", "Jump 2", "Jump 2", "Love!")]	Place 1 has Valentine's day. Place 3 has Valentine's day. Place 0 has Valentine's day. Place 2 has Valentine's day. Cupid's last position was 2. Mission was successful.	