JS Fundamentals Mid Exam Preparation

Problem 1 - Black Flag

Link: https://judge.softuni.org/Contests/Practice/Index/1773#0

Pirates are invading the sea, and you're tasked to help them plunder

Create a program that checks if **target plunder** is **reached**. First, you will receive how many **days** the pirating lasts. Then you will receive how much the pirates **plunder for a day**. Last you will receive the **expected plunder** at the end.

Calculate how much **plunder** the pirates manage to **gather**. Each **day** they gather the **plunder**. Keep in mind that they attack more ships every third day and add additional plunder to their total gain, which is **50% of the daily plunder**. Every **fifth day** the pirates encounter a warship, and after the battle, they **lose 30%** of their **total plunder**.

If the gained plunder is **more or equal** to the target, print the following:

`"Ahoy! {totalPlunder} plunder gained."`

If the gained plunder is **less** than the target. Calculate the **percentage left** and print the following:

`"Collected only {percentage}% of the plunder."`

Both numbers should be **formatted** to the **2nd decimal place**.

Input

- On the **1st line,** you will receive the **days** of the plunder – an **integer number** in the range [0…100000]
- On the **2nd line,** you will receive the **daily plunder** – an **integer number** in the range [0…50]
- On the **3rd line,** you will receive the **expected plunder** – a **real number** in the range [0.0…10000.0]

Output

- In the end, print whether the plunder **was successful** or **not,** following the format **described above**.

Examples

Input	Output

`(["5",`
`"40",`
`"100"])`
	`Ahoy! 154.00 plunder gained.`

Comments	
The days are 5, and the daily plunder is 40. On the third day, the total plunder is 120, and since it is a third day, they gain an additional 50% from the daily plunder, which adds up to 140. On the fifth day, the plunder is 220, but they battle with a warship and lose 30% of the collected cargo, and the total becomes 154. That is more than expected.	

`(["10",`
`"20",`
`"380"])`
	`Collected only 36.29% of the plunder.`

Problem 2 - Shopping List

Link: https://judge.softuni.org/Contests/Practice/Index/2031#1

It's the end of the week, and it is time for you to go shopping, so you need to create a shopping list first.

Input

You will receive an **initial list** with groceries separated by an exclamation mark **`"!"`**.

After that, you will be receiving **4 types** of commands until you receive **`"Go Shopping!"`**.

- **`"Urgent {item}"`** - **add** the item at the **start** of the list. If the item **already exists,** skip this command.
- **`"Unnecessary {item}"`** - **remove** the item with the given name, only **if it exists** in the list. Otherwise, skip this command.
- **`"Correct {oldItem} {newItem}"`** - if the item with the given **old name** exists, **change** its name with the **new** one. Otherwise, skip this command.
- **`"Rearrange {item}"`** - if the grocery exists in the list, **remove** it from its **current position** and **add** it at the **end** of the list. Otherwise, skip this command.

Constraints

- There won't be any duplicate items in the initial list

Output

- Print the **list** with all the groceries, joined by **`", "`**:
  `"{firstGrocery}, {secondGrocery}, … {nthGrocery}"`

Examples

Input	Output

`(["Tomatoes!Potatoes!Bread",`
`"Unnecessary Milk",`
`"Urgent Tomatoes",`
`"Go Shopping!"])`
	`Tomatoes, Potatoes, Bread`

Input	**Output**

`(["Milk!Pepper!Salt!Water!Banana",`
`"Urgent Salt",`
`"Unnecessary Grapes",`
`"Correct Pepper Onion",`
	`Milk, Onion, Salt, Water, Banana`

"Rearrange Grapes", "Correct Tomatoes Potatoes", "Go Shopping!"])	

Problem 3 - Moving Target

Link: https://judge.softuni.org/Contests/Practice/Index/2305#2

You are at the shooting gallery again, and you need a program that helps you keep track of moving targets. On the first line, you will receive a **sequence of targets with their integer values**, split by a **single space**. Then, you will start receiving **commands for manipulating the targets** until the **"End"** command. The commands are the following:

- **`"Shoot {index} {power}"`**
  - Shoot the target at the index **if it exists** by **reducing** its **value** by the **given power** (**integer value**).
  - Remove the target **if it is shot**. A target is considered **shot** when **its value reaches 0**.
- **`"Add {index} {value}"`**
  - Insert a target with the received value at the received **index if it exists**.
  - If not, print: **`"Invalid placement!"`**
- **`"Strike {index} {radius}"`**
  - **Remove** the target at the given **index** and **the ones before and after it** depending on the **radius**.
  - If **any of the indices** in the range is **invalid**, print: **`"Strike missed!"`** and **skip** this command.

Example: `"Strike 2 2"`

	{radius}	{radius}	{strikeIndex}	{radius}	{radius}		

- **`"End"`**
  - **Print** the sequence with targets in the following format and **end the program**:
    `"{target₁}|{target₂}…|{targetₙ}"`

Input / Constraints

- On the **first line,** you will receive **the sequence of targets** – **integer values [1-10000]**.
- On the **following lines,** until the **"End"** will be receiving the command described above – **strings**.
- There will never be a case when the **"Strike"** command would empty the whole sequence.

Output

- Print the appropriate message in case of any command if necessary.
- In the end, print the sequence of targets in the format described above.

Examples

Input	Output	Comments
(["52 74 23 44 96 110", "Shoot 5 10", "Shoot 1 80", "Strike 2 1", "Add 22 3", "End"])	Invalid placement! 52|100	The first command is "**Shoot**", so we reduce the target on **index 5**, which is valid, with the given **power** – **10**. Then we receive the same command, but we need to reduce the target on the 1st index, with power 80. The value of this target is 74, so it is considered shot, and we **remove** it. Then we receive the "**Strike**" command on the 2nd index, and we need to check if the range with radius 1 is valid:

		52 23 44 96 100 And it is, so we **remove** the targets. At last, we receive the "**Add**" command, but the index is **invalid**, so we print the appropriate **message**, and in the end, we have the following result: **52|100**

`(["1 2 3 4 5",`
`"Strike 0 1",`
`"End"])`
	`Strike missed!`
	`1|2|3|4|5`	