## Problem 1 – Counter Strike

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/2305#0)

Write a program that **keeps track of every won** battle against an **enemy**. You will receive **initial energy**. Afterward, you will start receiving the **distance** you need **to reach an enemy** until the **"End of battle"** command is given, or you **run out of energy**.

The **energy** you need for reaching an enemy is **equal to the distance you receive**. Each time you reach an enemy, you **win** a battle, and your **energy is reduced**. Otherwise, if you don't have **enough energy** to reach an enemy, **end the program** and **print**: **"Not enough energy! Game ends with {count} won battles and {energy} energy"**.

Every **third won battle** increases **your energy with the value of your current count of won battles**.

Upon receiving the **"End of battle"** command**,** print the **count of won battles** in the following format:

**"Won battles: {count}. Energy left: {energy}"**

### Input / Constraints

* On the **first line,** you will receive **initial energy** – an **integer [1-10000]**
* On the **following lines,** you will be receiving the **distance** of an enemy – an **integer** **[1-10000]**

### Output

* The description contains the proper output messages for each case and the format they should be printed.

### Examples

|  | **Input** | **Output** | **Comments** |
| --- | --- | --- | --- |
|  | 100  10  10  10  1  2  3  73  10 | Not enough energy! Game ends with 7 won battles and 0 energy | The initial energy is **100**. The first distance is **10**, so we subtract **10** from **100**, and we consider this a **won** battle. We are left with 90 energy. Next distance – **10**, and **80** energy left.  Next distance – **10**, **3** won battles and **70** energy, but since we have **3** won battles, we increase the energy with the current count of won battles, in this case – **3, and it becomes 73**.  The last distance we receive – **10** is unreachable since we have **0** energy, so we print the appropriate message, and the program ends. |
| 200  54  14  28  13  End of battle | | Won battles: 4. Energy left: 94 |  |

## Problem 2 – Treasure Hunt

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/4380/programming-fundamentals-with-csharp-january-2024)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/1773#1)

*The pirates need to carry a treasure chest safely 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, which is a **string** of **items** separated by a **"|"**.

**"{loot1}|{loot2}|{loot3} … {lootn}"**

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

* **"Loot {item1} {item2}…{itemn}"**:
  + 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 at 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 much** as there are.
  + Print the stolen items separated by **", "**:

**"{item1}, {item2}, {item3} … {itemn}"**

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 are going to receive the **initial treasure chest** (**loot** separated by **"|"**)
* On the following **lines**, until **"Yohoho!"**, you will be receiving commands.

### 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. |

## Problem 3 – Memory Game

A problem for exam preparation for the [Programming Fundamentals Course @ SoftUni](https://softuni.bg/trainings/4380/programming-fundamentals-with-csharp-january-2024)

Submit your solutions in the SoftUni Judge system [here](https://judge.softuni.org/Contests/Practice/Index/2517#2)

Write a program that recreates the **Memory game**.

On the first line, you will **receive a sequence of elements**. Each element in the sequence **will have a** **twin**. Until the player receives **"end"** from the console, you will receive **strings with two integers** separated by a space, representing **the indexes** of elements in the sequence.

If the player **tries to cheat** and enters **two equal indexes** or indexes which are **out of bounds of the sequence**, you should **add** two matching elements at the middle of the sequence in the following format:

**"-{number of moves until now}a"**

Then print this message on the console:

**"Invalid input! Adding additional elements to the board"**

### Input

* On the **first** line**,** you will receive a **sequence of elements.**
* On the **following** lines, you will receive **integers** until the command **"end".**

### Output

* Every time the player hit **two matching elements**, you should **remove** them from the sequence and **print** on the console the following message:

**"Congrats! You have found matching elements - ${element}!"**

* If the player hit **two different elements**, you should **print** on the console the following message:

**"Try again!"**

* If the player hit **all matching elements** before he receives **"end"** from the console, you should **print** on the console the following message:

**"You have won in {number of moves until now} turns!"**

* If the player receives **"end"** **before** **he hits all matching elements**, you should **print** on the console the following message:

**"Sorry you lose :(**

**{the current sequence's state}"**

### Constraints

* **All elements in the sequence will always have a matching element.**

### Examples

| **Input** | **Output** |
| --- | --- |
| 1 1 2 2 3 3 4 4 5 5  1 0  -1 0  1 0  1 0  1 0  end | Congrats! You have found matching elements - 1!  Invalid input! Adding additional elements to the board  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 3!  Congrats! You have found matching elements - -2a!  Sorry you lose :(  4 4 5 5 |
| **Comment** | |
| 1)  1 0  1 1 2 2 3 3 4 4 5 5 –> 1 = 1, equal elements, so remove them. Moves: 1  2)  -1 0  -1 is invalid index so we add additional elements  2 2 3 3 -2а -2а 4 4 5 5, Moves: 2  3)  1 0  2 2 3 3 -2а -2а 4 4 5 5 -> 2 = 2, equal elements, so remove them. Moves: 3  4)  1 0  3 3 -2а -2а 4 4 5 5 -> 3 = 3, equal elements, so remove them. Moves: 4  5)  1 0  -2а -2а 4 4 5 5 -> -2а = -2а, equal elements, so remove them. Moves: 5  6)  You receive the end command.  There are still elements in the sequence, so the player loses the game.  Final state - 4 4 5 5 | |
| a 2 4 a 2 4  0 3  0 2  0 1  0 1  end | Congrats! You have found matching elements - a!  Congrats! You have found matching elements - 2!  Congrats! You have found matching elements - 4!  You have won in 3 turns! |
| a 2 4 a 2 4  4 0  0 2  0 1  0 1  end | Try again!  Try again!  Try again!  Try again!  Sorry you lose :(  a 2 4 a 2 4 |