# Programming Fundamentals with Python: Exam Preparation

# Black Flag

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/1773#0**](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. |

# Shopping List

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2031#1**](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  Rearrange Grapes  Correct Tomatoes Potatoes  Go Shopping! | Milk, Onion, Salt, Water, Banana |

# Memory Game

**Submit your solutions in the SoftUni judge system at** [**https://judge.softuni.org/Contests/Practice/Index/2517#2**](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 |