# Python Advanced: Exam Preparation

## Temple of Doom

[**Link to Judge**](https://judge.softuni.org/Contests/Practice/Index/4081#0)

*Harry will have to discover an important artifact in a cursed temple. Relying on the inscriptions at the entrance, Harry realizes that he will have to face unprecedented challenges. He must take all the useful things that he has in his truck...*

There will be given **two sequences of integers,** representing **tools** and **substances** that he has at his disposal. There will also be a **third sequence of integers**, representing all **challenges** in the temple.

Your task is to **take the first tool from the tools sequence and the last substance from the substances sequence**. **Multiply the values** and **check the result**.

* If the calculated **result is equal to any of the elements from the challenge sequence**, the challenge is resolved. You need to **remove both the tool and the substance** from their sequences. The **challenge should** also **be removed** from its sequence.
* If the calculated **result is not equal to any of the elements from the challenge sequence**, the challenge is not resolved:
  + **Increase the value of the tool element by 1** and **move** the element **to the back** of the **tools’** sequence.
  + **Decrease** the value of the **substance** element **by 1** and **return the element** to the **substance’s sequence. If the value of the substance element reaches 0, remove it from the sequence.**

If Harry **has no substances or tools left** (the substances sequence is empty) **but has more challenges to resolve**, he is lost in the temple forever. End the program and print on the console the following message:

* **"Harry is lost in the temple. Oblivion awaits him."**

If Harry manages to **pass all the challenges**, he will find the artifact. End the program and **print on the console** the following message:

* **"Harry found an ostracon, which is dated to the 6th century BCE."**

## Input

* The first line will represent the **tools** that Harry has at his disposal – **integers**, separated by a **single space.**
* The second line will represent the **substances** that Harry has at his disposal – **integers**, separated by a **single space**.
* The third line will represent the **challenges** that Harry will have to resolve – **integers**, separated by a **single space**.

## Output

* On the **first** line **print on the console** the appropriate **message**, **among the following**:
  + **"Harry is lost in the temple. Oblivion awaits him."**
  + **"Harry found an ostracon, which is dated to the 6th century BCE."**
* On the next three lines, **print on the console** the elements of the **non-empty sequences**, in the following format:
  + **"Tools: element1, element2, element3 … elementn"**
  + **"Substances: element1, element2, element3 … elementn"**
  + **"Challenges: element1, element2, element3 … elementn"**

## Constraints

* All the given numbers will be valid **integers** in the range **[1, 100].**
* There will be **no negative inputs**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 4 6 8  11 3 5 7 9  24 28 18 30 | Harry found an ostracon, which is dated to the 6th century BCE.  Substances: 11 |
| **Comment** | |
| 1. 2 4 6 8   11 3 5 7 9  24 28 18 30  2 \* 9 = 18  Harry resolves one of the challenges. Remove both the tool value and the substance value from their sequences. Remove the challenge value from the challenges sequence.   1. 4 6 8   11 3 5 7  24 28 30  4 \* 7 = 28  Harry resolves one more of the challenges. Remove both the tool value and the substance value from their sequences. Remove the challenge value from the challenges sequence.   1. 6 8   11 3 5  24 30  6 \* 5 = 30  Harry resolves one more of the challenges. Remove both the tool value and the substance value from their sequences. Remove the challenge value from the challenges sequence.   1. 8   11 3  24  8 \* 3 = 24  Harry resolves one of the challenges. Remove both the tool value and the substance value from their sequences. Remove the challenge value from the challenges sequence.   1. The challenges sequence remains empty, so the program ends. The only sequence containing any element is the substances sequence, so it should be presented in the console output. | |
| **Input** | **Output** |
| 13 7 4 22 11 15 20  3 2 1  12 10 5 | Harry is lost in the temple. Oblivion awaits him.  Tools: 20, 14, 8, 5, 23, 12, 16  Challenges: 12, 10, 5 |
| **Comment** | |
| 1. 13 7 4 22 11 15 20   3 2 1  12 10 5  13 \* 1 = 13  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1. It becomes equal to 0, so it should be removed from the substance sequence.   1. 7 4 22 11 15 20 14   3 2  12 10 5  7 \* 2 = 14  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1.   1. 4 22 11 15 20 14 8   3 1  12 10 5  4 \* 1 = 4  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1. It becomes equal to 0, so it should be removed from the substance sequence.   1. 22 11 15 20 14 8 5   3  12 10 5  22 \* 3 = 66  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1.   1. 11 15 20 14 8 5 23   2  12 10 5  11 \* 2 = 22  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1.   1. 15 20 14 8 5 23 12   1  12 10 5  15 \* 1 = 15  No element from the challenges sequence is equal to the calculated result. Increase the tool value by 1 and add it to the back of the tools sequence. Decrease the substance value by 1. It becomes equal to 0, so it should be removed from the substance sequence.   1. 20 14 8 5 23 12 16   12 10 5   1. The substances sequence remains empty, so the program ends. The sequences containing any element are the tools sequence and the challenges sequence, so they should be presented in the console output. | |

## The Squirrel

[**Link to Judge**](https://judge.softuni.org/Contests/Practice/Index/3893#1)

*An intern from a big company must solve the game - "The Squirrel". He doesn’t have enough experience, so he needs your help.*

**Here are the rules of the game:**

The game starts with 0 collected hazelnuts. **Your goal is to collect 3 of them.**

You get as input the size of the field, which will be always a **square shape**. After that, you will receive the directions in which the squirrel can move – **"left"**, **"right"**, **"down"**, and **"up"** in a sequence, each value separated by a comma and a space (**", "**). On the next rows, you will receive the field.

Possible characters in the field:

* **s** - represents the squirrel's position.
* **h** – represents a hazelnut.
* **\*** – the asterisk represents an empty position.
* **t** – represents a trap.

The squirrel starts from the **s - s-position**.

* If the squirrel steps on a hazelnut, you have to **increase them by 1**. The position should be marked with an **asterisk** (**\***).
  + If the squirrel **collects all 3 hazelnuts**, the game ends.
* **Asterisk (\*) does nothing**, so nothing happens if the squirrel **steps on** it.
* If it steps on a trap, **the game ends**.
* If the squirrel moves **out** of the field, **the game ends**.

After all commands you will have 4 possible results:

* **You win if the squirrel collects 3 of the hazelnuts.**
* **The squirrel collects less than 3 hazelnuts.**
* **The squirrel steps on a trap.**
* **The squirrel moves out of the field.**

### Input

* **On the first line, you will receive the length of the field – an integer number in the range [3, 5].**
* **On the second line, you will receive the commands to move the squirrel – an array of strings separated by ", ".**
* **In the next N lines, you will receive the values for every row.**

### Output

* **On the first line:**
  + If the squirrel goes out of the field - "**The squirrel is out of the field.**".
  + If the squirrel steps on a trap - "**Unfortunately, the squirrel stepped on a trap...**".
  + If the squirrel hasn’t collected all the hazelnuts - "**There are more hazelnuts to collect.**".
  + If the squirrel has collected all hazelnuts - "**Good job! You have collected all hazelnuts!**".
* On the second line, print the **number of collected hazelnuts** - **"Hazelnuts collected: {hazelnutsCount}"**

### Constraints

* The size of the field will be between **[3,5]**.
* There could be **one** or **no trap** on the field.
* There will always be **3 hazelnuts on the field**.

### Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 5  left, left, up, right, up, up  \*\*h\*\*  t\*\*\*\*  \*h\*\*\*  \*h\*s\*  \*\*\*\*\* | Good job! You have collected all hazelnuts!  Hazelnuts collected: 3 | The squirrel moves 2 times to the left and collects its first hazelnut. After that collect the second one. Finally, with the last "**up**" command, the squirrel collects its final hazelnut. |
| 4  down, down, right, right  \*s\*h  \*\*\*h  \*\*\*t  h\*\*\* | Unfortunately, the squirrel stepped on a trap...  Hazelnuts collected: 0 |  |
| 4  down, down, right, right  h\*\*\*  \*\*\*h  \*s\*t  \*\*h\* | The squirrel is out of the field.  Hazelnuts collected: 0 |  |

## Pets Hotel

[**Link to Judge**](https://judge.softuni.org/Contests/Practice/Index/4089#2)

*You are a pet hotel owner.   
You need to organize the accommodation for the new-coming pets.*

Write a function called **accommodate\_new\_pets** that **receives information** about the available capacity of the hotel, the maximum weight allowed per pet, the pet types, and their weight, and **returns the result after the accommodation**. The function will receive a **different number of arguments**. The arguments will be passed as follows:

* The first argument will be the available capacity of your hotel - an **integer** in the **range [0, 50];**
* The second argument will be the maximum weight limit - a **float number** representing the pet’s maximum allowed weight;
* The following arguments will be the **tuples with two elements** - the **first** one is the **pet type (string)**, and the **second** one is **the pet weight** **(float)**;

After receiving the information and calling the function, the program should **start tracking the accommodation process**:

* Take the **pet type** from each tuple **successively** and **if you have enough capacity**, **accommodate it**, and proceed to the next one. Keep in mind that you will also need to track the **total number of pets** for **each pet type** you accommodate.
* If a pet’s **weight** exceeds the maximum weight limit, **ignore it**, and proceed to the next one.
* If the available **capacity is 0 (zero)**, **STOP accommodating**!
  + You are not supposed to check the weight of the unaccommodated pets (if any) when you run out of space.

In the end:

* If you’ve managed to **accommodate all pets**, return the message: **"All pets are accommodated! Available capacity: {available\_capacity}."**
* **Otherwise**, return the message: **"You did not manage to accommodate all pets!"**
* On the following lines **return** the accommodated **pet types** and **number of pets, ordered ascending (alphabetically) by pet type.** Each on a new line:

**"Accommodated pets:**

**{pet\_type1}: {number}**

**{pet\_type2}: {number}**

**…**

**{pet\_typeN}: {number}"**

***Note: Submit only the function in the judge system***

### Input

* There will be **no input from the console**, just parameters passed to your function.

### Output

* Return one of the **strings** **shown above** depending on the result **and the details** about accommodated pets **as described**.

### Constraints

* The **first** argument will always be an **integer**.
* The **second** argument will always be a **float** **number**.
* Each **tuple** given will always contain the **pet type** and **pet weight**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| print(accommodate\_new\_pets(  10,  15.0,  ("cat", 5.8),  ("dog", 10.0),  )) | All pets are accommodated! Available capacity: 8.  Accommodated pets:  cat: 1  dog: 1 |
| print(accommodate\_new\_pets(  10,  10.0,  ("cat", 5.8),  ("dog", 10.5),  ("parrot", 0.8),  ("cat", 3.1),  )) | All pets are accommodated! Available capacity: 7.  Accommodated pets:  cat: 2  parrot: 1 |
| print(accommodate\_new\_pets(  2,  15.0,  ("dog", 10.0),  ("cat", 5.8),  ("cat", 2.7),  )) | You did not manage to accommodate all pets!  Accommodated pets:  cat: 1  dog: 1 |