Python Advanced: Exam Preparation

1. Flower Finder

Link to Judge: https://judge.softuni.org/Contests/Practice/Index/3374#0

You will be given **two sequences of characters, representing vowels and consonants**. Your task is to start checking if the following words could be found:

- **`"rose"`**
- **`"tulip"`**
- **`"lotus"`**
- **`"daffodil"`**

Start by taking the **first character** of the **vowels collection** and the **last character** from the **consonants collection.** Then **check** if these letters are present in one or more of the given words. If a letter is present, that **part of the word** is considered **found**. The word is gradually revealed with each letter found. Continue processing the **next couple of letters** until you find **one of the given words above.**

A **letter (vowels or consonants) could participate in more than one word or more than one time in a word, for example:**

- The letter **`"o"`** is present in **`"rose"`**, **`"lotus"`**, and **`"daffodil"`**.
- The letter **`"l"`** is present in **`"tulip"`**, **`"lotus"`**, and **`"daffodil"`**.
- The letter **`"f"`** is present in the word **`"daffodil"`** twice.

The consonant and **the vowel** are **always removed** from the collection after trying to match them with the letters in the given words (whether successful or not). In the end, the program **stops** when **a word is found, or there are no more vowels or consonants**.

As a result, if you **found a word**, print **it** and **the remaining letters** in each collection in the format described below. Otherwise, print **`"Cannot find any word!"`** on the first line and **the remaining letters** in each sequence in the format described below.

Look at the provided examples for a better understanding of the problem.

Input

- On the **first line**, you will receive **vowels**, **separated** by a single space (**`" "`**).
- On the **second line**, you will receive **consonants, separated** by a single space (**`" "`**).

Output

- On the first line:
  - If a word is found, print it in the format: **`"Word found: {word_found}"`**
  - Otherwise, print: **`"Cannot find any word!"`**
- On the next lines, print the remaining letters in each collection (if there are any left):
  - **`"Vowels left: {vowel_one} {vowel_two} … {vowel_N}"`**
  - **`"Consonants left: {consonants_one} {consonants_two} … {consonants_N}"`**

Constraints

- All letters will be lowercase.
- The letter **`'y'` will always be a vowel.**
- The letter **`'w'` will always be a consonant.**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

Examples

Input	Output

`o e a o e a i`
`p r s x r`
	`Word found: rose`
	`Vowels left: o e a i`
	`Consonants left: p r`

Comment
Start by taking the first volew "o" and the last consonant "r". They are found in words "rose", "lotus", and "daffodil". Then, take "e" and "x". They are found in the word "rose". Then, take "a" and "s". They are found in words "rose", "lotus", and "daffodil". The word "rose" is found, so we print it. Then we print the remaining letters in each sequence.

Input	Output

`a a a`
`x r l t p p`
	`Cannot find any word!`
	`Consonants left: x r l`

`u a o i u y o e`
`p m t l`
	`Word found: tulip`
	`Vowels left: u y o e`

2. Martian Explorer

Link to Judge: https://judge.softuni.org/Contests/Practice/Index/3430#1

Your rover has landed on Mars, and it needs to find resources to start humanity's first interplanetary colony.

You will receive a **6x6 field on separate lines** with:

- **One rover** - marked with the letter **"E"**
- **Water deposit** (one or many) - marked with the letter **"W"**
- **Metal deposit** (one or many) - marked with the letter **"M"**
- **Concrete deposit** (one or many) - marked with the letter **"C"**
- **Rock** (one or many) - marked with the letter **"R"**
- **Empty positions** will be marked with **"-"**

After that, you will be given the **commands for the rover's movement** on **one line** separated by **a comma and a space** `(", ")`. Commands can be: **"up"**, **"down"**, **"left"**, or **"right"**.

For **each command**, the rover **moves in the given directions with one step**, and it can land on one of the given types of **deposit** or a **rock**:

- When it **lands on a deposit**, you must print the **coordinates of that deposit** in the format shown below and **increase its value by 1**.
- If the rover **lands on a rock**, it gets **broken.** Print the **coordinates where it got broken** in the format shown below, and **the program ends**.
- If the rover **goes out of the field**, it should **continue** from the **opposite side** in the **same direction**. Example: If the rover is at **position (3, 0)** and it needs to **move left** (outside the matrix), it should be placed at **position (3, 5)**.

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

The rover **needs to find at least one of each** deposit to **consider the area suitable** to start our colony.

Stop the program if you **run out of commands** or the **rover gets broken**.

Input

- On the **first 6 lines**, you will receive the **matrix**.
- On the **following line**, you will receive the commands for the rover **separated by a comma and a space.**

Output

- For each deposit found while you go through the commands, print out on the console: **`"{Water, Metal or Concrete} deposit found at ({row}, {col})"`**
- If the rover hits a rock, print the **coordinates where it got broken** in the format: **`"Rover got broken at ({row}, {col})"`**

After you go through all the commands or the rover gets broken, print out on the console:

- If the rover **has found at least one of each deposit**, print on the console: **`"Area suitable to start the colony."`**
- Otherwise, print on the console: **`"Area not suitable to start the colony."`**

See examples for more clarification.

Examples

Input	Output

`- R - - - -`
`- - - - - R`
`- E - R - -`
`- W - - - -`
`- - - C - -`
`M - - - - -`
`down, right, down, right, down, left, left, left`
	`Water deposit found at (3, 1)`
	`Concrete deposit found at (4, 3)`
	`Metal deposit found at (5, 0)`
	`Area suitable to start the colony.`

`R - - - - -`
`- - C - - -`
`- - - - M -`
`- - W - - -`
`- E - W - R`
`- - - - - -`
`up, right, down, right, right, right`
	`Water deposit found at (3, 2)`
	`Water deposit found at (4, 3)`
	`Rover got broken at (4, 5)`
	`Area not suitable to start the colony.`

`R - - - - -`
`- - C - - -`
`- - - - M -`
`C - M - R M`
`- E - W - -`
`- - - - - -`
`right, right, up, left, left, left, left, left`
	`Water deposit found at (4, 3)`
	`Metal deposit found at (3, 2)`
	`Concrete deposit found at (3, 0)`
	`Metal deposit found at (3, 5)`
	`Rover got broken at (3, 4)`
	`Area suitable to start the colony.`

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Follow us:

3. Naughty or Nice

Link to Judge: https://judge.softuni.org/Contests/Practice/Index/3306#2

Santa Claus is always watching and seeing if children are good or bad. Only the nice children get Christmas presents, so Santa Claus is preparing his list this year to check which child has been good or bad.

Write a function called **`naughty_or_nice_list`** which will **receive**

- A **list** representing Santa Claus' "Naughty or Nice" list full of kids' names
- A **different number of arguments** (strings) **and/or keywords** representing commands

The function should **sort** the kids in the given Santa Claus list **into 3 lists**: **`"Nice"`**, **`"Naughty"`**, and **`"Not found"`**.

The list holds a different number of kids - tuples containing two elements: **a counting number** (integer) at the **first** position and **a name** (string) at the **second** position.

For example: **`[(3, "Amy"), (1, "Tom"), (7, "George"), (3, "Katy")]`**.

Next, the function could receive **arguments and/or keywords.**

Each **argument** is a **command**. The commands could be the following:

- **`"{counting_number}-Naughty"`** - if there is **only one tuple in the given list** with the **same number**, **MOVE** the kid to a list with **NAUGHTY** kids and **remove it** from the Santa list. Otherwise, ignore the command.
- **`"{counting_number}-Nice"`** - if there is **only one tuple in the given list** with the **same number**, **MOVE** the kid to a list with **NICE** kids and **remove it** from the Santa list. Otherwise, ignore the command.

Each **keyword** holds a **key** with a **name** (string), and each **value** will be a string (**`"Naughty"`** or **`"Nice"`**):

- If there is **only one tuple** with the **same name**, **MOVE** the kid to a list with **NAUGHTY** or to the list with **NICE** kids depending on the **value in the keyword**. Then, **remove it** from the Santa list.
- Otherwise, ignore the command.

All **remaining tuples** in the given Santa's list are **not found kids**, and they should be **MOVED** to the **`"Not found"`** list.

In the end, **return the final lists, each on a new line as described below.**

<u>Note: Submit only the function in the judge system</u>

Input

- There will be **no input**. Just parameters passed to your function.

Output

- The function should **return strings with the names on each list on separate lines, if there are any, otherwise skip the line:**
  - **`"Nice: {name1}, {name2} … {nameN}"`**
  - **`"Naughty: {name1}, {name2} … {nameN}"`**
  - **`"Not found: {name1}, {name2} … {nameN}"`**

Examples

Test Code	Output

`print(naughty_or_nice_list(`
`    [`
`        (3, "Amy"),`
`        (1, "Tom"),`
`        (7, "George"),`
`        (3, "Katy"),`
`    ],`
`    "3-Nice",`
`    "1-Naughty",`
`    Amy="Nice",`
`    Katy="Naughty",`
`))`
	`Nice: Amy`
	`Naughty: Tom, Katy`
	`Not found: George`

`print(naughty_or_nice_list(`
`    [`
`        (7, "Peter"),`
`        (1, "Lilly"),`
`        (2, "Peter"),`
`        (12, "Peter"),`
`        (3, "Simon"),`
`    ],`
`    "3-Nice",`
`    "5-Naughty",`
`    "2-Nice",`
`    "1-Nice",`
`    ))`
	`Nice: Simon, Peter, Lilly`
	`Not found: Peter, Peter`

`print(naughty_or_nice_list(`
`    [`
`        (6, "John"),`
`        (4, "Karen"),`
`        (2, "Tim"),`
`        (1, "Merry"),`
`        (6, "Frank"),`
`    ],`
`    "6-Nice",`
`    "5-Naughty",`
`    "4-Nice",`
`    "3-Naughty",`
`    "2-Nice",`
`    "1-Naughty",`
`    Frank="Nice",`
`    Merry="Nice",`
`    John="Naughty",`
`))`
	`Nice: Karen, Tim, Frank`
	`Naughty: Merry, John`

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us: