Problem 1

Maria wants to make a firework show for the wedding of her best friend.

We should help her to make the perfect firework show.

First, you will be given **a sequence of integers representing firework effects**. Afterwards you will be given another **sequence of integers representing explosive power**.

You need to start from the **first firework effect** and try to mix it with the **last explosive power**. If the **sum** of their values is:

- **divisible by 3, but it is not divisible by 5** – **create Palm firework** and **remove both** materials
- **divisible by 5, but it is not divisible by 3** – **create Willow firework** and **remove both** materials
- **divisible by both 3 and 5** – **create Crossette firework** and **remove both** materials

Otherwise, **decrease** the value of the **firework effect by 1** and **move** it at the **end** of the sequence. Then, try to mix the same **explosive power** with the next **firework effect.**

If any value is **equal to or below 0,** you should **remove it** from the sequence **before trying to mix it with the other.**

When you have **successfully prepared enough fireworks for the show** or you have **no more firework punches** or **explosive power,** you need to stop mixing**.**

To make the perfect firework show, Maria needs **3 of each** of the **firework types.**

Input

- On the **first line**, you will receive the integers representing the **firework effects**, **separated** by **", "**.
- On the **second line**, you will receive the integers representing the **explosive power**, **separated** by **", "**.

Output

- On the **first** line, print:
  - if Maria **successfully prepared** the firework show: **"Congrats! You made the perfect firework show!"**
  - if Maria **failed** to prepare it: **"Sorry. You can't make the perfect firework show."**
- On the **second** line, print all firework effects left **if there are any**:
  - **"Firework Effects left: {effect1}, {effect2}, (…)"**
- On the **third** line, print all explosive fillings left **if there are any**:
  - **" Explosive Power left: {filling1}, {filling2}, (…)"**
- Then**,** you need to print **all** fireworks and the **amount you have of them**:
  - **"Palm Fireworks: {count}"**
  - **"Willow Fireworks: {count}"**
  - **"Crossette Fireworks: {count}"**

Constraints

- All the given numbers will be integers in the range **[-100, 100]**.
- There will be no cases with empty sequences.

Examples

Input	Output
5, 6, 4, 16, 11, 5, 30, 2, 3, 27 1, 13, 5, 3, -7, 32, 19, 3, 5, 7, 22	Congrats! You made the perfect firework show! Palm Fireworks: 4 Willow Fireworks: 3 Crossette Fireworks: 3
Comment	
1) 5 + 22 = 27 is devisible by 3 -> Palm Firework. Remove both. 2) 6 + 7 = 13 -> can't create firework. Firework effect should be decreased with 1 -> 5 and moved at the end 3) 4 + 7= 11 -> can't create firework. Firework effect should be decreased with 1 -> 3 and moved at the end 3) 16 + 7 = 23 -> can't create firework. Firework effect should be decreased with 1 -> 15 and moved at the end 4) 11 + 7 = 18 is devisible by 3 -> Palm Firework. Remove both. 5) 5 + 5 = 10 is devisible by 5 -> Willow Firework. Remove both. 6) 30 + 3 = 33 is devisible by 3 -> Palm Firework. Remove both. 7) 2 + 19 = 21 is devisible by 3 -> Palm Firework. Remove both. 8) 3 + 32 = 35 is devisible by 5 -> Willow Firework. Remove both. 9) (-7) is negative, so we remove it before mixing. 10) 27 + 3 = 30 is devisible by 5 and 3 -> Crossette Firework. Remove both. 11) 5 + 5 = 10 is devisible by 5 -> Willow Firework. Remove both. 12) 3 + 13 = 16 -> can't create firework. Firework effect should be decreased with 1 -> 2 and moved at the end 13) 15 + 13 = 28 -> can't create firework. Firework effect should be decreased with 1 -> 14 and moved at the end 14) 2 + 13 = 15 is devisible by 5 and 3 -> Crossette Firework. Remove both. 15) 1 + 14 = 15 is devisible by 5 and 3 -> Crossette Firework. Remove both. We have enough fireworks to make a firework show.	

Input	Output
-15, -8, 0, -16, 0, -22 10, 5	Sorry. You can't make the perfect firework show. Explosive Power left: 10, 5 Palm Fireworks: 0 Willow Fireworks: 0 Crossette Fireworks: 0
Comment	
After removing all the invalid integers, the firework effects's sequence is empty and the program ends.	

02. Collecting Coins

You are playing a game, and your goal is to collect 100 coins.

On the first line, you will be given a **number** representing the **size of the field** with a **square** shape. On the following few lines, you will be given the **field** with:

- **One player** - randomly placed in it and marked with the symbol **"P"**
- **Numbers** for coins placed at different positions of the field
- **Walls** marked with **"X"**

After the field state, you will be given **commands** for the **player's movement**. Commands can be: "**up**", "**down**", "**left**", "**right**". If the command is invalid, you should ignore it.

The player **moves** in the given **direction** with one **step for each command and collects all the coins that come across**. If he goes out of the field, he should **continue to traverse the field** from the **opposite side** in the **same direction**.

Note: He can go through the **same path many times**, but he can **collect the coins just once** (the first time).

There are only **two possible outcomes** of the game:

- The player **hits a wall**, **loses the game**, and **his coins are reduced to 50% and rounded down** to the next-lowest number.
- The player collects **at least 100 coins** and wins the game.

For more clarifications, see the examples below.

Input

- **A number** representing the size of the field (matrix NxN)
- **A matrix** representing the field (each position **separated by a single space**)
- On each of the following lines, you will get a move command.

Output

- If the player won the game, print: **"You won! You've collected {total_coins} coins."**
- If the player loses the game, print: "**Game over! You've collected {total_coins} coins.**"
- Collected coins have to be **rounded down** to the next-lowest number.
- The player's path as **cooridnates in lists on separate lines:**

```
"Your path:
[{row_position1}, {column_position1}]
[{row_position2}, {column_position2}]
…
[{row_positionN}, {column_positionN}]"
```

Constrains

- There will be no case in which less than 100 coins will be in the field
- All given numbers will be valid integers in the range [0, 100]

Examples

Input	Output
5 1 X 7 9 11 X 14 46 62 0 15 33 21 95 X P 14 3 4 18 9 20 33 X 0 left right right up up right	You won! You've collected 125 coins. Your path: [3, 0] [3, 4] [3, 0] [3, 1] [2, 1] [1, 1] [1, 2]
8 13 18 9 7 24 41 52 11 54 21 19 X 6 4 75 6 76 5 7 1 76 27 2 37 92 3 25 37 52 X 56 72 15 X 1 45 45 X 7 63 1 63 P 2 X 43 5 1 48 19 35 20 100 27 42 80 73 88 78 33 37 52 X 22 up down up left	Game over! You've collected 0 coins. Your path: [5, 2] [4, 2] [5, 2] [4, 2] [4, 1]

Problem 3

Maria is opening a cupcake shop. Help her manage her inventory to improve stock availability.

Write a function called **`stock_availability`** which receives:

- an inventory **list** of boxes with different kinds of cupcake flavours
- **`"delivery"`** or **`"sell"`** as **second** parameter
- there **might** or **might not** be any **other parameters – numbers or strings** at the end

In case of **`"delivery"`** to the shop was delivered new boxes with diferent kinds of **cupcakes**:

- You should **add** the **boxes** at the **end** of the inventory list
- There will be always at least one box delivered

In case of **`"sell"`** Maria has a client and she is selling different **boxes** with cupcakes:

- If there is a **number** as **another parameter**, it means that Maria has sold that many **boxes with cupcakes** and you should **remove them** from the **beginning** of the inventory list
- If there is/are string/s as **another parameter/s**, it means that Maria has sold **ALL cupcake boxes of the ordered flavour/s.** Beware that **not everything the buyer has ordered might be in stock**, so you should check if the order is valid.
- If there are **no other parameters**, it means that Maria has sold only **the first box** of cupcakes and you should **remove it** of the inventory list

For more clarifications, see the examples below.

Input

- There will be **no input**
- **Parameters** will be passed to your function

Output

- The function should **return a new inventory list**
- All **commands** will be **valid**

Examples

Test Code

`print(stock_availability(["choco", "vanilla", "banana"], "delivery", "caramel", "berry"))`
`print(stock_availability(["chocolate", "vanilla", "banana"], "delivery", "cookie","banana"))`
`print(stock_availability(["chocolate", "vanilla", "banana"], "sell"))`
`print(stock_availability(["chocolate", "vanilla", "banana"], "sell", 3))`
`print(stock_availability(["chocolate", "chocolate", "banana"], "sell", "chocolate"))`
`print(stock_availability(["cookie", "chocolate", "banana"], "sell", "chocolate"))`
`print(stock_availability(["chocolate", "vanilla", "banana"], "sell", "cookie"))`

Output

```
['choco', 'vanilla', 'banana', 'caramel', 'berry']
['chocolate', 'vanilla', 'banana', 'cookie', 'banana']
['vanilla', 'banana']
[]
['banana']
['cookie', 'banana']
['chocolate', 'vanilla', 'banana']
```