Problem 1 - Computer Store

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2517#0.

Write a program that **prints you a receipt** for your new computer. You will receive the **parts' prices (without tax)** until you receive what type of customer this is - **special** or **regular**. Once you receive the type of customer you should print the receipt.

The **taxes are 20%** of each part's price you receive.

If the customer is **special**, he has a 10% discount on the total price with taxes.

If a given price is not a positive number, you should print **`"Invalid price!"`** on the console and continue with the next price.

If the total price is equal to zero, you should print **`"Invalid order!"`** on the console.

Input

- You will receive numbers representing **prices (without tax)** until command **`"special"`** or **`"regular":`**

Output

- The receipt should be in the following format:

```
"Congratulations you've just bought a new computer!
Price without taxes: {total price without taxes}$
Taxes: {total amount of taxes}$
-----------
Total price: {total price with taxes}$"
```

Note: All prices should be displayed to the second digit after the decimal point! The discount is applied only on the total price. Discount is only applicable to the final price!

Examples

Input	Output
1050 200 450 2 18.50 16.86 special	Congratulations you've just bought a new computer! Price without taxes: 1737.36$ Taxes: 347.47$ ----------- Total price: 1876.35$
Comment	
1050 – valid price, total 1050 200 – valid price, total 1250 … 16.86 – valid price, total 1737.36 We receive **special**	

Price is positive number, so it is valid order Price without taxes is 1737.36 Taxes: 20% from 1737.36 = 347.47 Final price = 1737.36 + 347.47 = 2084.83 Additional 10% discount for special customers 2084.83 – 10% = 1876.35

Input	Output
1023 15 -20 -5.50 450 20 17.66 19.30 regular	Invalid price! Invalid price! Congratulations you've just bought a new computer! Price without taxes: 1544.96$ Taxes: 308.99$ ----------- Total price: 1853.95$
regular	Invalid order!

JS Examples

Input	Output
(['1050', '200', '450', '2', '18.50', '16.86', 'special'])	Congratulations you've just bought a new computer! Price without taxes: 1737.36$ Taxes: 347.47$ ----------- Total price: 1876.35$

Comment
1050 – valid price, total 1050 200 – valid price, total 1250 … 16.86 – valid price, total 1737.36 We receive **special** Price is positive number, so it is valid order Price without taxes is 1737.36 Taxes: 20% from 1737.36 = 347.47 Final price = 1737.36 + 347.47 = 2084.83 Additional 10% discount for special customers 2084.83 – 10% = 1876.35

Input	Output
(['1023', '15', '-20', '-5.50', '450', '20',	Invalid price! Invalid price! Congratulations you've just bought a new computer! Price without taxes: 1544.96$ Taxes: 308.99$ ----------- Total price: 1853.95$

'17.66', '19.30', 'regular'])	
(['regular'])	Invalid order!

Problem 2 - The Lift

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2517#1.

Write a program that **finds a place for the tourist on a lift.**

Every wagon should have **a maximum of 4 people on it**. If a wagon is full, you should direct the people to **the next one with space** available.

Input

- **On the first line,** you will receive **how many people** are waiting to get **on the lift**
- **On the second line**, you will receive the **current state of the lift separated by a single space: " "**.

Output

When there is no more available space left on the lift, or there are **no more people in the queue**, you should print on the console the final state of the lift's wagons separated by **" "** and one of the following messages:

- If there are no more people and the lift have empty spots, you should print:

  "The lift has empty spots!

  {wagons separated by ' '}"

- If there are still people in the queue and no more available space, you should print:

  "There isn't enough space! {people} people in a queue!

  {wagons separated by ' '}"

- If the lift is full and there are no more people in the queue, you should print only the wagons separated by **" "**

Examples

Input	Output
15 0 0 0 0	The lift has empty spots! 4 4 4 3
Comment	
First state - 4 0 0 0 -> 11 people left Second state – 4 4 0 0 -> 7 people left Third state – 4 4 4 0 -> 3 people left	
Input	**Output**
20 0 2 0	There isn't enough space! 10 people in a queue! 4 4 4
Comment	
First state - 4 2 0 -> 16 people left Second state – 4 4 0 -> 14 people left Third state – 4 4 4 -> 10 people left, but there're no more wagons.	

JS Examples

Input	Output
["15", "0 0 0 0 0"]	The lift has empty spaces! 4 4 4 3 0
Comment	
First state - 4 0 0 0 -> 11 people left Second state – 4 4 0 0 -> 7 people left Third state – 4 4 4 0 -> 3 people left	
Input	**Output**
["20", "0 2 0"]	There isn't enough space! 10 people in a queue! 4 4 4
Comment	
First state - 4 2 0 -> 16 people left Second state – 4 4 0 -> 14 people left Third state – 4 4 4 -> 10 people left, but there're no more wagons.	

Problem 3 - Memory game

Problem for exam preparation for the Programming Fundamentals Course @SoftUni.
Submit your solutions in the SoftUni judge system at https://judge.softuni.org/Contests/Practice/Index/2517#1.

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	Congrats! You have found matching elements - 1!

1 0 -1 0 1 0 1 0 1 0 end	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 -2a -2a 4 4 5 5, Moves: 2
3)
1 0
2 2 3 3 -2a -2a 4 4 5 5 -> 2 = 2, equal elements, so remove them. Moves: 3
4)
1 0
3 3 -2a -2a 4 4 5 5 -> 3 = 3, equal elements, so remove them. Moves: 4
5)
1 0
-2a -2a 4 4 5 5 -> -2a = -2a, 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

JS 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 - -1a! 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 -2a -2a 4 4 5 5, Moves: 2 3) 1 0 2 2 3 3 -2a -2a 4 4 5 5 -> 2 = 2, equal elements, so remove them. Moves: 3 4) 1 0 3 3 -2a -2a 4 4 5 5 -> 3 = 3, equal elements, so remove them. Moves: 4 5) 1 0 -2a -2a 4 4 5 5 -> -2a = -2a, 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