Problem 1 - Computer Store

Submit your solutions in the SoftUni judge system here

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 to 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	

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

Follow us:

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!`

Problem 2. Shoot for the Win

Submit your solutions in the SoftUni judge system here

Write a program that helps you keep track of your **shot targets**. You will receive a **sequence with integers**, separated by a single space, representing targets and their value. Afterward, you will be receiving indices until the **"End"** command is given, and you need to print the **targets** and the **count of shot targets**.

Every time you receive an **index**, you need to shoot the target on that index, **if it is possible**.

Every time you **shoot a target**, its value becomes **-1, and it is considered shot**. Along with that, you also need to:

- **Reduce** all the other **targets**, which have **greater values** than your **current** target, **with its value**.
- **Increase** all the other **targets**, which **have less than or equal** value to the **shot target**, **with its value.**

Keep in mind that you can't shoot a target, which is already shot. You also can't increase or reduce a target, which is considered a shot.

When you receive the **"End"** command, print the targets in their current state and the **count of shot targets** in the following format:

"Shot targets: {count} -> {target$_1$} {target$_2$}… {target$_n$}"

Input / Constraints

- On the **first line** of input, you will receive a **sequence** of **integers**, **separated** by **a single space – the targets sequence**.
- On the **following lines**, until the **"End"** command, you be receiving **integers** each on a single line – **the index of the target to be shot.**

Output

- The format of the output is described above in the problem description.

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

Examples

Input	Output	Comments

`24 50 36 70`
`0`
`4`
`3`
`1`
`End`
	`Shot targets 3 -> -1 -1 130 -1`	First, we shoot the target on index 0. It becomes equal to -1, and we start going through the rest of the targets. Since 50 is more than 24, we reduce it to 26 and 36 to 12 and 70 to 46. The sequence looks like that:
		-1 26 12 46
		The following index is invalid, so we don't do anything. Index 3 is valid, and after the operations, our sequence should look like that:
		-1 72 58 -1
		Then we take the first index with value 72, and our sequence looks like that:
		-1 -1 130 -1
		Then we print the result after the **"End"** command.

`30 30 12 60 54 66`
`5`
`2`
`4`
`0`
`End`
	`Shot targets: 4 -> -1 120 -1 66 -1 -1`	

Problem 3 - Memory Game

Submit your solutions in the SoftUni judge system here

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 -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

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

Follow us:

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`

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