Problem 1 - Counter-Strike

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/2305#0.

Write a program that **keeps track of every won** battle against an **enemy**. You will receive **initial energy**. Afterward, you will start receiving the **distance** you need **to reach an enemy** until the **"End of battle"** command is given, or you **run out of energy**.

The **energy** you need for reaching an enemy is **equal to the distance you receive**. Each time you reach an enemy, you **win** a battle, and your **energy is reduced**. Otherwise, if you don't have **enough energy** to reach an enemy, **end the program** and **print**: **"Not enough energy! Game ends with {count} won battles and {energy} energy"**.

Every **third won battle** increases **your energy with the value of your current count of won battles**.

Upon receiving the **"End of battle"** command**,** print the **count of won battles** in the following format:

"Won battles: {count}. Energy left: {energy}"

Input / Constraints

- On the **first line,** you will receive **initial energy** – an **integer [1-10000]**.
- On the **following lines,** you will be receiving the **distance** of an enemy – an **integer [1-10000]**

Output

- The description contains the proper output messages for each case and the format they should be printed.

Examples

Input	Output	Comments
100 10 10 10 1 2 3 73 10	Not enough energy! Game ends with 7 won battles and 0 energy	The initial energy is 100. The first distance is 10, so we subtract 10 from 100, and we consider this a **won** battle. We are left with 90 energy. Next distance – 10, and 80 energy left. Next distance – 10, 3 won battles and 70 energy, but since we have 3 won battles, we increase the energy with the current count of won battles, in this case – **3, and it becomes 73**. The last distance we receive – **10** is unreachable since we have **0** energy, so we print the appropriate message, and the program ends.
200 54 14 28 13 End of battle	Won battles: 4. Energy left: 94	

JS Examples

Input	Output	Comments
["100", "10", "10", "10", "1", "2", "3", "73", "10"])	Not enough energy! Game ends with 7 won battles and 0 energy	The initial energy is 100. The first distance is 10, so we subtract 10 from 100, and we consider this a **won** battle. We are left with 90 energy. Next distance – 10, and 80 energy left. Next distance – 10, 3 won battles and 70 energy, but since we have 3 won battles, we increase the energy with the current count of won battles, in this case – **3, and it becomes 73**. The last distance we receive – **10** is unreachable since we have **0** energy, so we print the appropriate message, and the program ends.
["200", "54", "14", "28", "13", "End of battle"])	Won battles: 4. Energy left: 94	

Problem 2 - Shoot for the Win

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 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.

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	Shot targets: 4 -> -1 120 -1 66 -1 -1	

4 0 End		

JS 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 next 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	

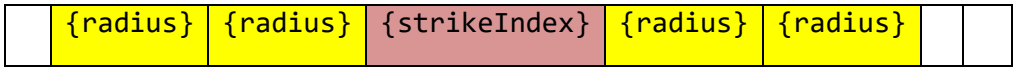
Problem 3 - Moving Target

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/2305#2.

You are at the shooting gallery again, and you need a program that helps you keep track of moving targets. On the first line, you will receive a **sequence of targets with their integer values**, split by a **single space**. Then, you will start receiving **commands for manipulating the targets** until the **"End"** command. The commands are the following:

- **"Shoot {index} {power}"**
  - Shoot the target at the index **if it exists** by **reducing** its **value** by the **given power** (**integer value**).
  - Remove the target **if it is shot**. A target is considered **shot** when **its value reaches 0**.
- **"Add {index} {value}"**
  - Insert a target with the received value at the received **index if it exists**.
  - If not, print: **"Invalid placement!"**
- **"Strike {index} {radius}"**
  - **Remove** the target at the given **index** and **the ones before and after it** depending on the **radius**.
  - If **any of the indices** in the range is **invalid**, print: **"Strike missed!"** and **skip** this command.

  Example: "Strike 2 2"



- **"End"**
  - **Print** the sequence with targets in the following format and **end the program**:
    "{target$_1$}|{target$_2$}…|{target$_n$}"

Input / Constraints

- On the **first line,** you will receive **the sequence of targets** – **integer values [1-10000]**.
- On the **following lines,** until the **"End"** will be receiving the command described above – **strings**.
- There will never be a case when the **"Strike"** command would empty the whole sequence.

Output

- Print the appropriate message in case of any command if necessary.
- In the end, print the sequence of targets in the format described above.

Examples

Input	Output	Comments
52 74 23 44 96 110 Shoot 5 10 Shoot 1 80 Strike 2 1 Add 22 3 End	Invalid placement! 52|100	The first command is "**Shoot**", so we reduce the target on **index 5**, which is valid, with the given **power** – **10**. Then we receive the same command, but we need to reduce the target on the 1st index, with power 80. The value of this target is 74, so it is considered shot, and we **remove** it. Then we receive the "**Strike**" command on the 2nd index, and we need to check if the range with radius 1 is valid:

		52 23 44 96 100 And it is, so we **remove** the targets. At last, we receive the "**Add**" command, but the index is **invalid**, so we print the appropriate **message**, and in the end, we have the following result: **52|100**
1 2 3 4 5 Strike 0 1 End	Strike missed! 1|2|3|4|5	

JS Examples

Input	Output	Comments
["52 74 23 44 96 110", "Shoot 5 10", "Shoot 1 80", "Strike 2 1", "Add 22 3", "End"])	Invalid placement! 52|100	The first command is "**Shoot**", so we reduce the target on **index 5**, which is valid, with the given **power – 10**. Then we receive the same command, but we need to reduce the target on the 1st index, with power 80. The value of this target is 74, so it is considered shot, and we **remove** it. Then we receive the "**Strike**" command on the 2nd index, and we need to check if the range with radius 1 is valid: **52 23 44 96 100** And it is, so we **remove** the targets. At last, we receive the "**Add**" command, but the index is **invalid**, so we print the appropriate **message**, and in the end, we have the following result: **52|100**
["1 2 3 4 5", "Strike 0 1", "End"])	Strike missed! 1|2|3|4|5	