Programming Fundamentals with Python: Exam Preparation

1. Bonus Scoring System

Submit your solutions to the SoftUni Judge system.

Create a program that calculates **bonus points** for each **student** enrolled in a course. On the **first** line, you are going to receive **the number of students**. On the **second** line, you will receive **the total number of lectures** in the course. The course has **an additional bonus**, which you will receive **on the third line**. On the following lines, you will be receiving the **count of attendances for each student**.

The bonus is calculated with the following **formula**:

```
{total bonus} = {student attendances} / {course lectures} * (5 + {additional bonus})
```

Find the student with the **maximum bonus** and print them, along with **his attendance,** in the following format:

```
"Max Bonus: {max bonus points}."
"The student has attended {student attendances} lectures."
```

Round the bonus points at the end to **the nearest larger number**.

Input / Constraints

- On the **first line,** you are going to receive the **number of the students** – an integer in the range [0…50]
- On the **second line,** you will receive the **number of the lectures** – an integer number in the range [0...50].
- On the **third line**, you will receive **the additional bonus** – an integer number in the range [0….100].
- **On the following lines**, you will be receiving the **attendance of each student**.
- There will **never** be **students with equal bonuses**.

Output

- Print the **maximum bonus points** and the **attendances** of the given student, **rounded** to the nearest **larger** number, scored by a student in this course in the format described above.

Examples

Input	Output
5 25 30 12 19 24 16 20	Max Bonus: 34. The student has attended 24 lectures.

Comments	
First, we receive the **number of students** enrolled in the course – **5**. The total count of the lectures is **25,** and the additional bonus is **30**. Then we calculate the bonus of the student with 12 attendances, which is **16.8**. We continue calculating **each of the student's bonuses**. The one **with 24 attendances** has the **highest bonus – 33.6 (34 rounded)**, so we print the appropriate message on the console.	
10 30 14 8 23 27 28 15 17 25 26 5 18	Max Bonus: 18. The student has attended 28 lectures.

2. Mu Online

Submit your solutions to the SoftUni Judge system.

You have **initial health 100 and initial bitcoins 0**. You will be given **a string representing the dungeon's rooms**. Each room is separated with **'|'** (vertical bar): **"room1|room2|room3…"**

Each room contains **a command** and a **number**, separated by space. The command can be:

- **"potion"**
    - You are healed with the number in the second part. But your health **cannot exceed** your **initial health (100)**.
    - First print: **"You healed for {amount} hp."**
    - After that, print your current health: **"Current health: {health} hp."**
- **"chest"**
    - You've found some bitcoins, the number in the second part.
    - Print: **"You found {amount} bitcoins."**
- In **any other case,** you are **facing a monster**, which you will **fight**. The **second part of the room** contains the **attack** of the monster. You should remove the monster's attack from your health.
    - If you are not dead (health <= 0), you've slain the monster, and you should print: **"You slayed {monster}."**

- If you've died, print **"You died! Killed by {monster}."** and your quest is over. Print the best room you've manage to reach: **"Best room: {room}"**

If you managed to **go through all the rooms** in the dungeon, print on the **following three lines**:

"You've made it!"

"Bitcoins: {bitcoins}"

"Health: {health}"

Input / Constraints

You receive a **string** representing the dungeon's rooms, separated with **'|'** (vertical bar): **"room1|room2|room3…"**.

Output

Print the corresponding messages described above.

Examples

Input	Output
rat 10|bat 20|potion 10|rat 10|chest 100|boss 70|chest 1000	You slayed rat. You slayed bat. You healed for 10 hp. Current health: 80 hp. You slayed rat. You found 100 bitcoins. You died! Killed by boss. Best room: 6
Input	**Output**
cat 10|potion 30|orc 10|chest 10|snake 25|chest 110	You slayed cat. You healed for 10 hp. Current health: 100 hp. You slayed orc. You found 10 bitcoins. You slayed snake. You found 110 bitcoins. You've made it! Bitcoins: 120 Health: 65

3. Man O War

Submit your solutions to the SoftUni [Judge system](https://judge.softuni.org/).

The pirates encounter a huge Man-O-War at sea.

Create a program that **tracks** the **battle** and either chooses a **winner** or prints a **stalemate**. On the **first line,** you will receive the **status** of the **pirate ship**, which is a **string** representing **integer sections** separated by **">"**. On **the second line,** you will receive the **same** type of status, but for the **warship**:

"{section₁}>{section₂}>{section₃}… {sectionₙ}"

On the **third line,** you will receive the **maximum health capacity** a section of the ship can reach.

The following lines represent commands **until "Retire"**:

- **"Fire {index} {damage}"** - the pirate ship **attacks** the warship with the **given damage** at that section. Check if the **index is valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the warship **sinks**, print the following and **stop** the program: **"You won! The enemy ship has sunken."**
- **"Defend {startIndex} {endIndex} {damage}"** - the warship **attacks** the pirate ship with the **given damage** at that **range** (**indexes are inclusive)**. Check if both **indexes are valid** and if not, **skip** the command. If the section **breaks** (health <= 0) the pirate ship **sinks**, print the following and **stop** the program:
"You lost! The pirate ship has sunken."
- **"Repair {index} {health}"** - the crew **repairs** a section of the **pirate ship** with the **given health**. Check if the **index is valid** and if not, **skip** the command. The health of the section **cannot** exceed the **maximum health capacity**.
- **"Status"** - prints the **count** of all sections of the **pirate ship** that need repair soon, which are all sections that are **lower than 20%** of the **maximum health capacity**. Print the following:
"{count} sections need repair."

In the end, if a **stalemate** occurs, print the **status** of **both** ships, which is the **sum** of their individual sections, in the following format:

"Pirate ship status: {pirateShipSum}

Warship status: {warshipSum}"

Input

- On the **1st line,** you are going to receive the **status** of the **pirate ship** (**integers** separated by **'>'**)
- On the **2nd line,** you are going to receive the **status** of the **warship**
- On the **3rd line,** you will receive the **maximum health** a section of a ship can reach.
- On the following **lines**, until **"Retire"**, you will be receiving commands.

Output

- Print the output in the **format described above**.

Constraints

- The **section numbers** will be integers in the range [**1….1000**]
- The **indexes** will be integers [**-200….200**]
- The **damage** will be an integer in the range [**1….1000**]
- The **health** will be an integer in the range [**1….1000**]

Examples

Input	Output

`12>13>11>20>66`
`12>22>33>44>55>32>18`
`70`
`Fire 2 11`
`Fire 8 100`
`Defend 3 6 11`
`Defend 0 3 5`
`Repair 1 33`
`Status`
`Retire`
	`2 sections need repair.`
	`Pirate ship status: 135`
	`Warship status: 205`

Comments

First, we receive the command "`Fire 2 11`", and damage the warship at section index 2, which is currently 33, and after reduction, the status of the warship is the following:

`12 22 22 44 55 32 18`

The **second** and **third** commands have **invalid indexes**, so we skip them.

The **fourth** command, **"`Defend 0 3 5`"** damages **4 sections** of the pirate ship with **5,** which results in the following states:

`7 8 6 15 66`

The **fifth** command, **"`Repair 1 33`"** repairs the pirate ship section and adds **33 health** to the current **8,** which results in **41**

Only **2 sections** of the pirate ship (**7** and **6**) need repair soon.

In the end, there is a **stalemate,** so we print both ship statuses (**sum** of all sections).

Input	Output

`2>3>4>5>2`
`6>7>8>9>10>11`
`20`
`Status`
`Fire 2 3`
`Defend 0 4 11`
`Repair 3 18`
	`3 sections need repair.`
	`You lost! The pirate ship has sunken.`

`Retire`	