02. Dragon Accounting

One of the oldest professions after prostitution is accounting. It dates back to the dragon era. When dragons started trading goods for eggs, they felt the need to establish the eggs as a de facto currency. The Dragon Karl Marx founded the first company ever, where other dragons worked for **money** (eggs). But he had hard times running his own company's accounting and needs some help. He invented his own **rules** for accounting and used this chart of accounts.

- Previous years deficit – ***expense***
- Machines – ***expense***
- Product development - ***income***
- Taxes – ***expense***
- Unconditional funding – ***income***

The boss also **pays salaries** every 30 days. Assuming all 30 days are paydays, he pays salaries **proportionally** (e.g. an employee is hired on **20th day**, when **30th day** hits, he receives salary for **10** days. Daily salary should be **computed** to the 9th digit after the decimal separator up and then kept to the 7th digits after the decimal point without any rounding). Formula for calculating total salary to receive is employee's salary per day multiplied by the days the employee has worked that month. **((salary / 30) * totalWorkingDaysThatMonth)**

On the first line of input you will be given the initial capital of the company. Each of the next lines is in format **{hired};{fired};{salary};{additional events: *money*},** for instance:

40;10;800;Machines:100;Taxes:50;Product development:300.

This means that 40 employees are **hired,** 10 are **fired,** the **salary** of the new employees is **800, 100** is paid for **machines, 50** is paid for **taxes, 300** is received from **Product development.** The employee starts working immediately.

Firing employees is done starting from the oldest hired. E.g. if the company hired 5 workers on the first day and on the **next day** he hires 2 and fires 4, he will fire 4 from the initial 5 and end up with 2 new employees and 1 old. Then, if he hires another 2 and fires 2 more, he will fire the **only remaining** employee from the initial 5 and 1 from the first 2 employees hired.

See the following example. At the end of day **3**, we have **0** employees from day **1, 1** employee from day **2** with **150** salary, and **2** employees from day **3** with **251.18** salary.

Current day: →	1	2	3
Employees hired on day: ↓	Hired: 5, Fired: 0, Salary: 500.5	Hired: 2, Fired: 4, Salary: 150	Hired: 2, Fired: 2, Salary 251.18
1	Count: 5 Salary: 500.5 Days: 1	Count: 1 Salary: 500.5 Days: 2	Count: 0 Salary: 500.5 Days: 3
2	0	Count: 2 Salary: 150 Days: 1	Count: 1 Salary: 150 Days: 2
3	0	0	Count: 2 Salary: 251.18 Days: 1

When day 30 hits, the only employee from day 2 would have worked 29 days; total salary: **(150 / 30)* 29**. The salary of the two employees from day 3 would be (**251.18 / 30) * 28.** This formula should be used even when the employee has worked full month. **(Salary / 30) * 30.**

There are no employees left from day 1, so no money would be taken from your budget.

All daily activities need to finish before you check for bankruptcy (**giving salaries**, calculating **income/expenses**…). The order of operations is **hire employees** -> **check for raise -> give salaries -> fire employees -> check for additional income/expense -> check for bankruptcy.**

You need to keep track of the company's current state, and, if the company has **bankrupted**, stop receiving input and print on the console "**BANKRUPTCY: {amountOfMoneyCompanyInDebt}**". When you receive "**END**", print on the console "**{employeesLeft} {capital}".** {**Capital**}, and {**amountOfMoneyCompanyInDebt}** are to be printed to the fourth digit after the decimal point, without any rounding e.g. **10** is **10.0000, 12.5** is **12.5000, 13.456789** is **13.4567.**

When an employee stays for more than a year in the company he/she receives a **0.6% increase** in salary. The next year he/she will receive **another** raise, if not fired. Assume that a year has **exactly 365** days and a **month** has **30** days.

Input

- On the first line, the initial capital is given
- On each next line, string in format **{hired};{fired};{salary};{additional events:*money*}** is received
- Input finishes when the command "END" is reached

Output

- There is one line of output
- In case of bankruptcy **- BANKRUPTCY: {amountOfMoneyCompanyInDebt}**
- If the program finishes normally, print **{employeesLeft} {capital}**
- All money outputs must consist of 4 digits after the decimal separator without rounding and the decimal separator must be ".".

Constraints

- There will be **no** more than 2,000 lines of input
- **Hired** or **fired** employees will be valid integers between 0 and 3.000.000.000
- The **initial capital**, **income/expense** and **salaries** will be valid decimal numbers between negative infinity and positive infinity.
- The reasons for spending or receiving money will be **valid** strings from the chart of accounts
- You cannot fire more employees than you currently have

Examples

Input	Output
12345.1234567 10;0;40;Machines:120.4 7;6;44;Product development:8.8;Taxes:12 END	11 12221.5234
12345.1234567	BANKRUPTCY: 111234.5999

10;0;40;Machines:120.4 7;6;44;Product development:8.8;Taxes:12 4;4;123456789;Machines:123456.1234 12;3;100;Unconditional funding:10000000000000.0000 1;2;18;Machines:4