Exercises: Communication, Events and Reflection Exercises

This document defines the exercises for "C# OOP Advanced" course @ Software University. Please submit your solutions (source code) of all below described problems in Judge.

Problem 1. *Mirror Image

You are given Mirror images, wizards, reflections... or not.

Basically, you have a Wizard which has an **id**, **name** and **magical power**. The id of the initial wizard is always zero.

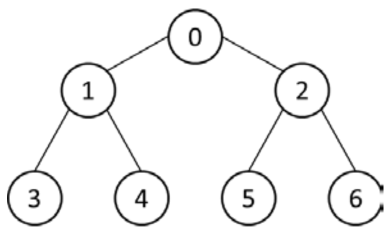
He can cast **two spells** - **Reflection** and **Fireball**.

Reflection creates two mirror images of the wizard with **ids representing the instance of a reflection being created starting from 1**. A mirror image has half the magical power of its creator. Every wizard/mirror image **cannot have more than two own mirror images**.

Fireball creates a ball of fire which deals damage equal to its caster's magical power.

When a wizard casts a spell, all of his images cast the same spell after him. The same is true for the mirror images.

Consider the picture below.

Picture	Explanation
	If the initial wizard (number 0) casts Reflection two times you will first end up with mirror images 1 and 2, then with 3, 4 followed by 5 and 6. After that, if wizard 0 casts Fireball, the sequence in which his mirror images will cast fireball after him would be 1, 3, 4 and then 2, 5 and lastly 6.

Create a program which by a given id and a spell name makes the corresponding wizard or mirror image cast the spell.

Input

On the first line you will get the name and magical power of the initial wizard. Next the input will come as commands from the console and will be one of the following, until an "END" command is given:

- **"<id> REFLECTION"** - The wizard casts a Reflection spell
- **"<id> FIREBALL"** - The wizard casts a Fireball spell
- **"END"** - Program execution stops

Output

The output should be printed as follows:

- For Reflection spell - **"<name> <id> casts Reflection"**
- For Fireball spell - **"<name> <id> casts Fireball for <magical power> damage"**

Constraints

- The id of the initial wizard always starts from zero.
- Every magical power division is made on integers, so half the magical power of 11 is 5.

Examples

Input	Output

`Oz 12`
`0 FIREBALL`
`0 REFLECTION`
`0 REFLECTION`
`3 FIREBALL`
`4 FIREBALL`
`5 FIREBALL`
`6 FIREBALL`
`END`
	`Oz 0 casts Fireball for 12 damage`
	`Oz 0 casts Reflection`
	`Oz 0 casts Reflection`
	`Oz 1 casts Reflection`
	`Oz 2 casts Reflection`
	`Oz 3 casts Fireball for 3 damage`
	`Oz 4 casts Fireball for 3 damage`
	`Oz 5 casts Fireball for 3 damage`
	`Oz 6 casts Fireball for 3 damage`

Problem 2. *1984

The institutions are always watching you.

You are given three different entities - **Employees**, **Companies** and **Institutions**. Employees have **name** and **income**. Companies have **name**, **turnover** and **revenue**. Employees and Companies have an **id** which is unique and never changes. Everything else can change.

Institutions have **name** and one or more **subjects which they are monitoring**. Every institution always watches if a company or an employee has changed some of their attributes and saves the change if it is within its subject.

For example, an Institution named "NAP" monitors "income". If an Employee changes its income, the Nap Institution saves this change.

A change should have an information about the entity that has changed (e.g. employee or company), the type of the changed property (e.g. string or double), the name of the changed property (e.g. income or name), the old value and the new value.

Save all changes in format **"--<entity>(ID:<id>) changed <field name>(<field type>) from <old value> to <new value>"**. For example, **"--Employee(ID:1) changed name(String) from Gosho to Misho".**

Input

The first line consists of 3 numbers - **M**, **N** and **P**.

M - the number of entities, **N** - the number of institutions and **P** - the number of changes entities will make.

- On the next M lines, you will get the information about all entities in one of the following formats **"Employee <Id> <Name> <Income>"** for employees or **"Company <Id> <Name> <Turnover> <Revenue>"** for companies.
- On the next N lines, you will get the information about all institutions in the format **"Institution <Id> <Name> <subject 1> <subject 2> .. <subject n>".**
- On the next P lines, you will get an id which changes some of its fields in the format **"<Id> <Field name> <New value>"**

Output

Print all Institution logs starting with the name of the institution and the number of saved changes:

"<Institution name>: <Number of changes> changes registered"

"--<entity type>(ID:<id>) changed <field name>(<field type>) from <old value> to <new value>"

For example:

"NAP: 1 changes registered"

"--Employee changed int field "income" from 1200 to 1300"

Print the change logs of all Institutions in the order of their appearance.

Constraints

- Id is unique, cannot be changed and should be saved in a string variable.
- Name can be changed and should be saved in a string variable.
- All number fields (income, turnover and revenue) can be changed and should be saved in an int variables.
- The input will always be valid and in the format described. You will never receive a non-existent attribute to monitor, field name to change or a non-existent id.

Examples

Input	Output

`2 3 2`
`Employee 1 Gosho 900`
`Company 2 Magazin 5000 1000`
`Institution 3 MVR name`
`Institution 4 NAP turnover`
`Institution 5 DANS income`
`1 name Misho`
`2 turnover 6000`
	`MVR: 1 changes registered`
	`--Employee(ID:1) changed name(String) from Gosho to Misho`
	`NAP: 1 changes registered`
	`--Company(ID:2) changed turnover(int) from 5000 to 6000`
	`DANS: 0 changes registered`

`2 3 4`
`Employee 1 Gosho 900`
`Company 2 Magazin 5000 1000`
`Institution 3 MVR name turnover`
`Institution 4 NAP turnover`
`Institution 5 DANS income`
	`MVR: 3 changes registered`
	`--Employee(ID:1) changed name(String) from Gosho to Misho`
	`--Company(ID:2) changed name(String) from Magazin to BookShop`
	`--Company(ID:2) changed turnover(int) from 5000 to 4000`

`1 name Misho`
`2 name BookShop`
`2 turnover 4000`
`1 income 1200`
	`NAP: 1 changes registered`
	`--Company(ID:2) changed turnover(int) from 5000 to 4000`
	`DANS: 1 changes registered`
	`--Employee(ID:1) changed income(int) from 900 to 1200`

Hint

Try using Observer pattern, reflection, annotations and generics for a cleaner solution.