Working with Abstraction

Problem 1. Raw Data

You should refactor Pesho's working code so that it isn't using just one class.

You are the owner of a courier company and want to make a system for tracking your cars and their cargo. Define a class **Car** that holds information about **model, engine, cargo** and a **collection of exactly 4 tires**. The **engine**, **cargo** and **tire** should be **separate classes**. Create a constructor that receives all information about the **Car** and creates and initializes its inner components (engine, cargo and tires).

On the first line of input you will receive a number **N** - the amount of cars you have. On each of the next **N** lines you will receive information about a car in the format "**<Model> <EngineSpeed> <EnginePower> <CargoWeight> <CargoType> <Tire1Pressure> <Tire1Age> <Tire2Pressure> <Tire2Age> <Tire3Pressure> <Tire3Age> <Tire4Pressure> <Tire4Age>"** where the speed, power, weight and tire age are **integers**, tire pressure is a **double.**

After the **N** lines you will receive a single line with one of 2 commands: "**fragile**" or "**flamable**". If the command is "**fragile**" print all cars whose **Cargo Type is "fragile"** with **a tire** whose **pressure is < 1**. If the command is "**flamable**" print all cars whose **Cargo Type is "flamable"** and have **Engine Power > 250**. The cars should be printed in order of appearing in the input.

Examples

Input	Output
2 ChevroletAstro 200 180 1000 fragile 1.3 1 1.5 2 1.4 2 1.7 4 Citroen2CV 190 165 1200 fragile 0.9 3 0.85 2 0.95 2 1.1 1 fragile	Citroen2CV
4 ChevroletExpress 215 255 1200 flamable 2.5 1 2.4 2 2.7 1 2.8 1 ChevroletAstro 210 230 1000 flamable 2 1 1.9 2 1.7 3 2.1 1 DaciaDokker 230 275 1400 flamable 2.2 1 2.3 1 2.4 1 2 1 Citroen2CV 190 165 1200 fragile 0.8 3 0.85 2 0.7 5 0.95 2 flamable	ChevroletExpress DaciaDokker

Problem 2. Cars Salesman

Here Pesho has done pretty well actually. Your job, however, is to show him how to reuse base constructors in his classes.

Define two classes **Car** and **Engine.** A **Car** has a **model, engine, weight** and **color**. An Engine has **model**, **power, displacement** and **efficiency**. A Car's **weight** and **color** and its Engine's **displacements** and **efficiency** are **optional**.

On the first line you will read a number **N** which will specify how many lines of engines you will receive. On each of the next **N** lines you will receive information about an **Engine** in the following format "<**Model> <Power> <Displacement> <Efficiency>**". After the lines with engines, on the next line you will receive a number **M** – specifying the number of Cars that will follow. On each of the next **M** lines information about a **Car** will follow in the format "<**Model> <Engine> <Weight> <Color>**", where the engine will be the **model of an existing Engine**. When

creating the object for a **Car**, you should keep a **reference to the real engine** in it, instead of just the engine's model. Note that the optional properties **might be missing** from the formats.

Your task is to print each car (in the order you received them) and its information in the format defined bellow, if any of the optional fields has not been given print "**n/a**" in its place instead:

```
<CarModel>:
  <EngineModel>:
    Power: <EnginePower>
    Displacement: <EngineDisplacement>
    Efficiency: <EngineEfficiency>
  Weight: <CarWeight>
  Color: <CarColor>
```

Bonus*

Override the classes' **ToString**() methods to have a reusable way of displaying the objects.

Examples

Input	Output

2
V8-101 220 50
V4-33 140 28 B
3
FordFocus V4-33 1300 Silver
FordMustang V8-101
VolkswagenGolf V4-33 Orange
	FordFocus:
	  V4-33:
	  Power: 140
	  Displacement: 28
	  Efficiency: B
	  Weight: 1300
	  Color: Silver
	FordMustang:
	  V8-101:
	  Power: 220
	  Displacement: 50
	  Efficiency: n/a
	  Weight: n/a
	  Color: n/a
	VolkswagenGolf:
	  V4-33:
	  Power: 140
	  Displacement: 28
	  Efficiency: B
	  Weight: n/a
	  Color: Orange

4
DSL-10 280 B
V7-55 200 35
DSL-13 305 55 A+
V7-54 190 30 D
4
	FordMondeo:
	  DSL-13:
	  Power: 305
	  Displacement: 55
	  Efficiency: A+
	  Weight: n/a

FordMondeo DSL-13 Purple VolkswagenPolo V7-54 1200 Yellow VolkswagenPassat DSL-10 1375 Blue FordFusion DSL-13	Color: Purple VolkswagenPolo: V7-54: Power: 190 Displacement: 30 Efficiency: D Weight: 1200 Color: Yellow VolkswagenPassat: DSL-10: Power: 280 Displacement: n/a Efficiency: B Weight: 1375 Color: Blue FordFusion: DSL-13: Power: 305 Displacement: 55 Efficiency: A+ Weight: n/a Color: n/a

Problem 3. Jedi Galaxy

The next four problems are from Pesho's preparations for the exam. The code is working, but it doesn't look good. Rework his solutions in order to avoid repeating code, increase readability and improve performance (perhaps).

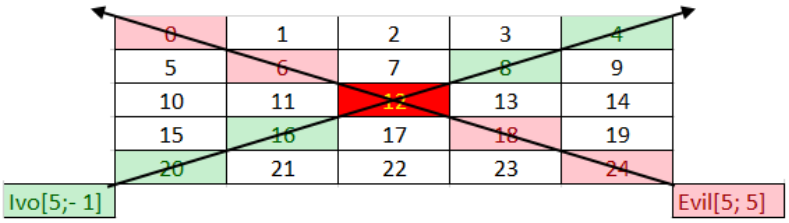
Ivo is the illegal son of Darth Vader. But he doesn't know much about being a powerful Jedi warrior. Meanwhile, Princess Lea just broke up with Han Solo, because he cheated on her. Ivo decided to grab her heart, but he needs your help. He must be powerful enough to impress her and so he starts gathering stars to grow stronger.

His galaxy is represented as a two-dimensional array. Every cell in the matrix is a star that has a value. Ivo starts at the given **col** and **row**. He can move only on the diagonal **from the lowest left to the upper right**, and **adds** to his score all the stars (values) from the cells he **passes through**. Unfortunately, there is always an Evil power that tries to prevent his success.

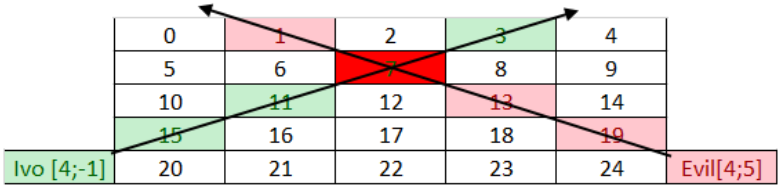
Evil power starts at the given **row** and **col** and instantly destroys all stars on the opposite diagonal – **From lowest right to the upper left.**

Ivo **adds** the values only of the stars that are **not destroyed** by the evil power.

You will receive **two** integers, separated by space, which represent the two dimensional array - the first being the rows and the second being the columns. Then, you must fill the two dimensional array with increasing integers starting from 0, and continuing on every row, like this:
first row: 0, 1, 2… m
second row: n+1, n+2, n+3… n + n.
Example:



Ivo starts with coordinates row = 5, col = -1. He must collect all stars with value [20, 16, 12, 8, 4]. Evil starts with coordinates row = 5, col = 5. The Evil destroys all stars in range [24, 18, 12, 6, 0]. The star with value **12** is the cross point for Ivo and The Evil, so Ivo skips the stars and collects only these who are not in the evil range.



You will also receive multiple pairs of commands in the form of 2 integers separated by a single space. The first two integers will represent Ivo's start coordinates. The second one will represent the Evil Power's start coordinates.

The input ends when you receive the command "**Let the Force be with you**". When that happens, you must print the value of all stars that Ivo has collected successfully.

Input

- On the first line, you will receive the number N, M -> the dimensions of the matrix. You must then fill the matrix according to these dimensions.
- On the next several lines you will begin receiving 2 integers separated by a single **space**, which represent Ivo's row and col. On the next line you will receive the Evil Power's coordinates.
- There will always be **at least 2 lines** of input to represent at least 1 path of Ivo and the Evil force.
- When you receive the command, "**Let the Force be with you**" the input ends.

Output

- The output is simple. Print the sum of the values from all stars that Ivo has collected.

Constraints

- The dimensions of the matrix will be integers in the range [5, 2000].
- The given rows will be valid integers in the range [0, 2000].
- The given columns will be valid integers in the range [$-2^{31} + 1$, $2^{31} - 1$].

Input	Output
5 5 5 -1 5 5 Let the Force be with you	48
5 5 4 -1 4 5 Let the Force be with you	29

Problem 4. Hospital

Your task will be to prepare an electronic register for a hospital. In the hospital we have different departments:

- Cardiology
- Oncology
- Emergency department
- etc.

Each department has **20** rooms for patients and **each room has 3 beds**. When a new patient goes in the hospital, he/she is placed on the first free bed in the department. If there are no free beds, the patient should go in another hospital. Of course, in every hospital there are doctors. Each doctor can have patients in a different department. You will receive information about patients in the format **{Department} {Doctor} {Patient}**

After the "Output" command you will receive some other commands with what kind of output you need to print. The commands are:

- **{Department}** – You need to **print all patients** in this department in the **order of receiving**
- **{Department} {Room}** – You need to **print all patients** in this room in **alphabetical order**
- **{Doctor}** – you need to **print all patients** for this doctor in **alphabetical order**

The program ends when you receive command "End".

Input

On the first lines you will receive info for the hospital, department, doctors and patients in the following format:

{Department} {Doctor} {Patient}

When you read the "**Output**" line you will get one or more commands telling you what you need to print

Read commands for printing, 'till you reach the command "**End**"

Output

- **{Department}** – print all patients in this department in order of receiving on new line
- **{Department} {Room}** – print all patients in this room in alphabetical order each on new line
- **{Doctor}** – print all patients that are healed from doctor in alphabetical order on new line

Constraints

- **{Department}** – single word with length **1 < n < 100**
- **{Doctor}** – name and surname, both with length **1 < n < 20**
- **{Patient}** – unique name with length **1 < n < 20**
- **{Room}** – integer **1 <= n <= 20**
- Time limit: 0.3 sec. Memory limit: 16 MB.

Examples

Input	Output

Cardiology Petar Petrov Ventsi Oncology Ivaylo Kenov Valio Emergency Mariq Mircheva Simo Cardiology Genka Shikerova Simon Emergency Ivaylo Kenov NuPogodi Cardiology Gosho Goshov Esmeralda Oncology Gosho Goshov Cleopatra Output Cardiology End	Ventsi Simon Esmeralda

Input	Output
Cardiology Petar Petrov Ventsi Oncology Ivaylo Kenov Valio Emergency Mariq Mircheva Simo Cardiology Genka Shikerova Simon Emergency Ivaylo Kenov NuPogodi Cardiology Gosho Goshov Esmeralda Oncology Gosho Goshov Cleopatra Output Cardiology 1 End	Esmeralda Simon Ventsi

Input	Output
Cardiology Petar Petrov Ventsi Oncology Ivaylo Kenov Valio Emergency Mariq Mircheva Simo Cardiology Genka Shikerova Simon Emergency Ivaylo Kenov NuPogodi Cardiology Gosho Goshov Esmeralda Oncology Gosho Goshov Cleopatra Output Ivaylo Kenov End	NuPogodi Valio

Problem 5. Greedy Times

Finally, you have unlocked the safe and reached the treasure! Inside there are all kinds of gems, cash in different currencies and gold bullions. Next to you there is a bag which unfortunately has a limited space. You don't have

much time so you need take as much wealth as possible! But in order to get bigger amount of the most valuable items you need to keep the following rules:

- The **gold amount** in your bag should **always be more** than **or equal** the **gem amount** at **any** time
- The **gem amount** should **always be more** than **or equal** the **cash amount** at **any** time

If you read an **item** which **breaks this rule** you **should not put** it in the **bag**. You should **always** be careful **not** to **exceed** the overall **bag's capacity** as it will tear down you will lose everything! You will receive the **content of** the **safe** on a **single line** in the **format** of **item - quantity** pairs separated by **whitespace**. You need to gather **only three types** of items:

- Cash - All **three letter** items
- Gem - All **items** which **end** on "**Gem**" (at least 4 symbols)
- Gold - this type has **only one item** with the name - "**Gold**"

Each **item** which **does not** fall **in** one of the **above categories** is **useless** and you should **skip it**. Reading item's **names** should be **CASE-INSENSITIVE**. You should **aggregate item's quantities** which have the **same name**.

If you kept the rules you should have escaped successfully with a bag full of wealth. Now it's time to review what you have managed to get out of the safe. **Print all** the **types** ordered by **total amount** in **descending order**. Inside a type **order** the **items** first **alphabetically** in **descending** order and **then by** their **amount** in **ascending** order. Use the following format for each type:

"<{type}> ${total amount}"

"##{item} - {amount}" - each item from this type on new line

Input

- On the **first line**, you will receive a **number** which represents the **capacity** of the **bag**
- On the **second line**, you will receive a **sequence** of **item - quantity** pairs

Output

Print **only** the **types** from which you **have items in the bag** ordered by **Total Amount** descending. Inside a type order the **items first alphabetically** in **descending** order and **then** by **amount** in **ascending order**. Use the following format for each type:

"<{type}> ${total amount}"

"##{item} - {amount}" - each item on new line

Constraints

- Bag's **max capacity** will **always** be a **positive number**
- All **quantities** will be **positive integer** in the range [0 … 2100000000]
- Each item of type **gem** will have a **name** - **at least 4** symbols
- Time limit: 0.1 sec. Memory limit: 16 MB

Examples

Input	Output
150 Gold 28 Rubygem 16 USD 9 GBP 8	<Gold> $28 ##Gold - 28

	<Gem> $16 ##Rubygem - 16 <Cash> $9 ##USD - 9
24000010 USD 1030 Gold 300000 EmeraldGem 900000 Topazgem 290000 CHF 280000 Gold 10000000 JPN 10000 Rubygem 10000000 KLM 3120010	<Gold> $10300000 ##Gold - 10300000 <Gem> $10290000 ##Topazgem - 290000 ##Rubygem - 10000000 <Cash> $3410010 ##KLM - 3120010 ##JPN - 10000 ##CHF - 280000
80345 RubyGem 70000 JAV 10960 Bau 60000 Gold 80000	<Gold> $80000 ##Gold - 80000

Problem 6. Sneaking

After our hero Sam got the recipe from the first problem, there is another thing he needs to check off from his to-do list. In order to make the recipe even more valuable, he needs to "eliminate" anyone who possesses the knowledge of it. That person is Sam's sworn enemy - **Nikoladze**. Sam needs to get through a rectangular room of **patrolling enemies** until he finally **reaches Nikoladze**.

A standard room looks like this:

Room	Legend

`......N...`
`b.........`
`..d.......`
`......d...`
`.....S....`
	S ➔ Sam, the player character
	b/d ➔ left/right-facing patrolling enemy
	N ➔ Nikoladze
	. ➔ Empty space

Each turn proceeds as follows:

- **Enemies** move either **left** or **right**, depending on which **direction** they are **facing** (**b** goes **right**, **d** goes **left**)
  - If an enemy is standing on the **edge** of the room, he flips his **direction** (from **d** to **b** or from **b** to **d**)
- After that, Sam moves in the **direction** he is instructed to (either **U/D/L/R** or **W**, which means **wait**).
- If **Sam** moves **onto an enemy** (**same row** and **column**), Sam **kills** the enemy and **leaves no trace of him**.
- Otherwise, if an enemy is on the **same row** as Sam, and also **facing Sam** (eg. **`.b.S.`**), the **enemy kills Sam**.
- If Sam is reaches the **same row** as **Nikoladze**, **Sam** kills **Nikoladze** (replacing him with an **X**)

Input

- On the **first line** of input, you will receive n – the **number of rows** the **room** will consist of
- On the next **n lines**, you will receive the **room**, which Sam will have to navigate.

- On the **final line** of input, you will receive a sequence of **directions** – one of (U, D, L, R, W)

Output

- If Sam is **killed**, print "**`Sam died at {row}, {col}`**"
- If Nikoladze is **killed**, print "**`Nikoladze killed!`**"
- Then, in both cases, **print** the **final state of the room** on the **console**, with either **Sam** or **Nikoladze's symbols** replaced by an **X**.

Constraints

- The room will always be **rectangular**.
- There will **always** be enough moves for **Sam** to reach **Nikoladze**
- There will be **no case** where **Sam** is instructed to move **out of the bounds of the room**.
- There will be **no case** with **two enemies on the same row**.
- There will be **no case** with an **enemy and Nikoladze** standing on the **same row**.
- There will be **no case** where Sam reaches the same **row and column** as **Nikoladze**.

Examples

Input	Output	Comments

5
`......N...`
`b.........`
`..d.......`
`......d...`
`.....S....`
UUUUR
	`Sam died at 2, 5`
	`......N...`
	`...b......`
	`b....X....`
	`..........`
	`..........`
		Turn 1: Enemies move, then Sam **steps on** the enemy on the **4th** row.
		Turn 2: Enemies move, then Sam moves.
		Turn 3: Enemy 2 **turns around**, Sam goes on the same row as him.
		Turn 4: Enemy **sees Sam** and **kills him**.

3
`N......`
`.b.....`
`..dS...`
WUUU
	`Nikoladze killed!`
	`X..S...`
	`.......`
	`b......`
		Turn 1: Enemies move, Sam waits.
		Turn 2: Enemies move, Sam goes **up**, **steps on an enemy**.
		Turn 3: Enemies move, Sam goes **up**, **kills Nikoladze**.

6
`............`
`....S.......`
`.b..........`
`..........d.`
`............`
`....N.......`
WWWDWWWDDRD
	`Nikoladze killed!`
	`............`
	`............`
	`...........b`
	`d...........`
	`............`
	`....XS......`
		Turn 1/2/3: Enemies **move**, Sam **waits**.
		Turn 4: Enemies **move**, Sam goes **down**.
		Turn 5/6/7: Enemies **move**, Sam **waits**.
		Turn 8/9: Enemies **move**, Sam goes **down**.
		Turn 10: Enemies **move**, Sam goes **right**.
		Turn 11: Enemies **move**, Sam goes **down** and **kills Nikoladze**.

Problem 7. Family Tree

Pesho is done with C# Advanced and is now working with classes. He waited for the last day the exercise is due and wrote some really bad code in a hurry. This time, his code doesn't even work well and he needs help to refactor and fix it to get max points in Judge.

You want to build your family tree, so you went to ask your grandmother. Sadly, your grandmother keeps remembering information about your predecessors in pieces, so it falls to you to group the information and build the family tree.

On the first line of input you will receive either a name or a birthdate in the format "**<FirstName> <LastName>"** or **"day/month/year**" – your task is to find the person's information in the family tree. On the next lines until the

command "**End**" is received you will receive information about your predecessors that you will use to build the family tree.

The information will be in one of the following formats:

- "**FirstName LastName - FirstName LastName**"
- "**FirstName LastName - day/month/year**"
- "**day/month/year - FirstName LastName**"
- "**day/month/year - day/month/year**"
- "**FirstName LastName day/month/year**"

The first 4 formats reveal a family tie – **the person on the left is parent to the person on the right** (as you can see the format does not need to contain names, for example the 4th format means the person born on the left date is parent to the person born on the right date). The last format ties different information together – i.e. **the person with that name was born on that date**. **Names** and **birthdates** are **unique** – there won't be 2 people with the same name or birthdate, there will **always** be enough entries to construct the family tree (all people's names and birthdates are known and they have atleast one connection to another person in the tree).

After the command "**End**" is received you should print all information about the person whose name or birthdate you received on the first line – his **name, birthday, parents and children** (check the examples for the format). The people in the parents and childrens lists should be ordered by their first appearance in the input (regardless if they appeared as a birthdate or a name, for example in the first input Stamat is before Penka because he has appeared first on the second line, while she appears on the third one).

Examples

Input	Output
Pesho Peshev 11/11/1951 - 23/5/1980 Penka Pesheva - 23/5/1980 Penka Pesheva 9/2/1953 Pesho Peshev - Gancho Peshev Gancho Peshev 1/1/2005 Stamat Peshev 11/11/1951 Pesho Peshev 23/5/1980 End	Pesho Peshev 23/5/1980 Parents: Stamat Peshev 11/11/1951 Penka Pesheva 9/2/1953 Children: Gancho Peshev 1/1/2005
13/12/1993 25/3/1934 - 4/4/1961 Poncho Tonchev 25/3/1934 4/4/1961 - Moncho Tonchev Toncho Tonchev - Lomcho Tonchev Moncho Tonchev 13/12/1993 Lomcho Tonchev 7/7/1995 Toncho Tonchev 4/4/1961 End	Moncho Tonchev 13/12/1993 Parents: Toncho Tonchev 4/4/1961 Children:
Asen Peshev 3/10/1927 - 23/10/1951 Gosho Peshev 3/10/1927 3/4/1923 - Maxim Peshev Maxim Peshev 26/8/1951 3/10/1927 - Stamat Peshev Stamat Peshev 24/6/1948 Dragomir Peshev - 5/2/1998	Asen Peshev 3/4/1923 Parents: Children: Maxim Peshev 26/8/1951 Kalin Peshev 23/10/1951

Goran Peshev 5/2/1998 Maxim Peshev - 4/3/1977 Dragomir Peshev 4/3/1977 3/10/1927 - Karamfil Peshev Karamfil Peshev 6/3/1951 Dragomir Peshev - 16/12/1998 Dimitrichka Pesheva 16/12/1998 3/4/1923 - Kalin Peshev Kalin Peshev 23/10/1951 Dragomir Peshev - 10/2/1999 Pesho8 Peshev 10/2/1999 Dragomir Peshev - 23/11/2001 Todor Peshev 23/11/2001 Dragomir Peshev - 18/8/2000 Kamen Peshev 18/8/2000 Dragomir Peshev - 21/6/2002 Trendafil Peshev 21/6/2002 Asen Peshev 3/4/1923 4/3/1977 - Stefan Peshev Stefan Peshev 2/6/2001 End	