01. Christmas Elves

Everything in the Satna Claus' workshop was going well until, on one freezing Sunday, a dangerous storm destroyed almost all toys. Now Santa's elves fear they won't be able to meet their December deadline. It could be a disaster, and some children around the world may not get their Christmas toys. Luckily, you've come up with an idea, and you just need to write a program that manages your plan.

The Christmas elves have special toy-making skills - each elf can make a toy from a given number of materials.

First, you will receive a sequence of **integers** representing each **elf's energy**. On the following line, you will be given another sequence of **integers,** each representing a **number of materials in a box**.

Your task is to calculate **the total elves' energy used** for making toys and **the total number of successfully made toys**.

You are very clever and have immediately recognized the **pros and cons of the work process** - the **first elf** takes **the last box of materials** and tries to create the toy:

- Usually, the elf **needs energy equal to the number of materials**. If he **has enough** energy, he **makes the toy**. His energy **decreases** by the used energy, and the **toy goes straight to Santa's bag**. Then, the elf **eats** a cookie reward which **increases his energy by 1**, and **goes to the end of the line**, preparing for the upcoming boxes.
- Every **third time** one of the elves **takes a box**, he tries his best to be creative, and **he will need twice as much energy as usual**. If **he has enough**, he manages to create **2 toys**. Then, his **energy decreases;** he eats a **cookie reward** and **goes to the end of the line,** similar to the first bullet.
- Every **fifth time** one of the elves **takes a box**, he is a little clumsy and somehow manages to **break the toy when he just made it (if he made it)**. The **toy is thrown away,** and the **elf doesn't get a cookie reward**. However, his **energy is already spent,** and it needs to be **added** to the total elves' energy.
    - **If an elf creates 2 toys, but he is clumsy, he breaks them.**
- If an elf does **not** have enough energy, he **leaves the box of materials to the next elf**. Instead of making the toy, the elf drinks a hot chocolate which **doubles his energy**, and **goes to the end of the line**, preparing for the upcoming boxes.

Note: North Pole's social policy is very tolerant of the elves. If the **current elf's energy** is **less than 5 units**, he **does NOT TAKE a box**, but he takes a day off. **Remove the elf** from the collection.

Stop crafting toys when you are out of materials or elves.

Input

- The **first** line of input will represent each elf's energy - **integers**, separated by a **single space**
- On the **second** line, you will be given the **number of materials in each box** - **integers**, separated by a **single space**

Output

- On the **first line**, print **the number of created toys**: **`"Toys: {total_number_of_toys}"`**
- On the **second line**, print **the total used energy**: **`"Energy: {total_used_energy}"`**
- On the **next two lines** print the **elves** and **boxes** that are **left**, **if there are any**, **otherwise skip the line:**
  - **`"Elves left: {elf1}, {elf2}, … {elfN}"`**
  - **`"Boxes left: {box1}, {box2}, … {boxN}"`**

Constraints

- All the elves' values will be **integers** in the range **[1, 100]**
- All the boxes' values will be **integers** in the range **[1, 100]**

Examples

Input	Output	Comment
10 16 13 25 12 11 8	Toys: 3 Energy: 31 Elves left: 3, 6, 26, 14	1) The elf with energy 10 takes the box with 8 materials. He creates **1 gift** and uses **8 units of energy**. He eats a cookie and goes to the end of the line, which now looks like this: **16 13 25 3**. 2) The elf with energy **16** takes the box with 11 materials. He creates **1 gift** and uses **11 units of energy**. Then, he eats a cookie and goes to the end of the line, which now looks like this: **13 25 3 6**. 3) The elf with energy **13** takes the box with **12 materials**. It is the **third** time an elf takes a box. The elf does not have the needed energy: **12 * 2**, so he drinks a hot chocolate and goes to the end of the line: **25 3 6 26**. 4) The elf with energy **25 takes the box** with 12 materials. It is the **fourth** time an elf takes a box. He creates **1 gift** and uses **12 units of energy**. He eats a cookie and goes to the end of the line, which now looks like this: **3 6 26 14**. No boxes are left, so the program ends. Print the desired text.
10 14 22 4 5 11 16 17 11 1 8	Toys: 7 Energy: 75 Elves left: 10, 14	

5 6 7 2 1 5 7 5 3	Toys: 3 Energy: 20 Boxes left: 2, 1	

02. North Pole Challenge

You are visiting Santa Claus' workshop, and it is complete chaos. You will need to help Santa find all items scattered around the workshop.

You will be given the size of the matrix in the format **`"{rows}, {columns}"`**. It is the **Santa's workshop** represented as some **symbols** separated by a **single space**:

- **Your position is** marked with the symbol **`"Y"`**
- **Christmas decorations are** marked with the symbol **`"D"`**
- **Gifts** are marked with the symbol **`"G"`**
- **Cookies** are marked with the symbol **`"C"`**
- **All of the empty positions** will be marked with **`"."`**

After the field state, you will be given **commands** until you receive the command **`"End"`**. The commands will be in the format **`"right/left/up/down-{steps}"`**. You should **move** in the given **direction** with the given **steps and collect all the items that come across**. If you go out of the field, you should **continue to traverse the field** from the **opposite side** in the **same direction**. You should mark your path with **`"x"`**. Your current position should always be marked with **`"Y"`**.

Keep track of all collected items. If you've collected all items at any point, **end the program** and **print**: **`"Merry Christmas!"`**.

Input

- On the first line, you will receive the number of rows and columns in the format **`"{rows}, {columns}"`**
- On the next **lines**, you will receive **each row with its columns - symbols**, separated by a single space.
- On the following **lines**, you will receive **commands** in the format described above until you receive the command **`"End"`** or until you **collect all items**.

Output

- On the **first line, if you have** collected all items, print:
  - **`"Merry Christmas!"`**
  - **Otherwise, skip the line**
- Next, print the **number of collected items** in the format:
  - **`"You've collected:`**
  - **`- {number_of_decoration} Christmas decorations`**
  - **`- {number_of_gifts} Gifts`**
  - **`- {number_of_cookies} Cookies"`**
- Finally, print the **field,** as shown in the examples.

Constrains

- All the **commands** will be **valid**
- There will **always be at least one item**
- The **dimensions** of the matrix will be integers in the range **`[1, 20]`**
- The field will always have **only one "Y"**
- **On the field, there will always be only the symbols shown above.**

Examples

Input	Output
6, 5 . . . . . C . . G . . C . . . G . . C . . D . . D Y . . . G left-3 up-1 left-2 right-7 up-1 End	You've collected: - 2 Christmas decorations - 1 Gifts - 0 Cookies . . . . . C . . G . . C . . . G . Y C . x x x x x x . x x x
5, 6 . . . . Y C D D . . . . . C . . . . . C . . right-3 down-3	Merry Christmas! You've collected: - 2 Christmas decorations - 0 Gifts - 3 Cookies . . . . x x x x . . . . . x . . . . . Y . .
5, 2 . . . . . Y . . . G up-1 left-11 down-10 End	You've collected: - 0 Christmas decorations - 0 Gifts - 0 Cookies x . Y x x x x . x G

03. Naughty or Nice

Santa Claus is always watching and seeing if children are good or bad. Only the nice children get Christmas presents, so Santa Claus is preparing his list this year to check which child has been good or bad.

Write a function called **`naughty_or_nice_list`** which will **receive**

- A **list** representing Santa Claus' "Naughty or Nice" list full of kids' names
- A **different number of arguments** (strings) **and/or keywords** representing commands

The function should **sort** the kids in the given Santa Claus list **into 3 lists**: **`"Nice"`**, **`"Naughty"`**, and **`"Not found"`**.

The list holds a different number of kids - tuples containing two elements: **a counting number** (integer) at the **first** position and **a name** (string) at the **second** position.

For example: **`[(3, "Amy"), (1, "Tom"), (7, "George"), (3, "Katy")]`**.

Next, the function could receive **arguments and/or keywords.**

Each **argument** is a **command**. The commands could be the following:

- **`"{counting_number}-Naughty"`** - if there is **only one tuple in the given list** with the **same number**, **MOVE** the kid to a list with **NAUGHTY** kids and **remove it** from the Santa list. Otherwise, ignore the command.
- **`"{counting_number}-Nice"`** - if there is **only one tuple in the given list** with the **same number**, **MOVE** the kid to a list with **NICE** kids and **remove it** from the Santa list. Otherwise, ignore the command.

Each **keyword** holds a **key** with a **name** (string), and each **value** will be a string (**`"Naughty"`** or **`"Nice"`**):

- If there is **only one tuple** with the **same name**, **MOVE** the kid to a list with **NAUGHTY** or to the list with **NICE** kids depending on the **value in the keyword**. Then, **remove it** from the Santa list.
- Otherwise, ignore the command.

All **remaining tuples** in the given Santa's list are **not found kids**, and they should be **MOVED** to the **`"Not found"`** list.

In the end, **return the final lists, each on a new line as described below.**

Note: Submit only the function in the judge system

Input

- There will be **no input**. Just parameters passed to your function.

Output

- The function should **return strings with the names on each list on separate lines, if there are any, otherwise skip the line:**

- "Nice: {name1}, {name2} … {nameN}"
- "Naughty: {name1}, {name2} … {nameN}"
- "Not found: {name1}, {name2} … {nameN}"

Examples

Test Code	Output

`print(naughty_or_nice_list(`
`  [`
`  (3, "Amy"),`
`  (1, "Tom"),`
`  (7, "George"),`
`  (3, "Katy"),`
`  ],`
`  "3-Nice",`
`  "1-Naughty",`
`  Amy="Nice",`
`  Katy="Naughty",`
`))`
	`Nice: Amy`
	`Naughty: Tom, Katy`
	`Not found: George`

`print(naughty_or_nice_list(`
`  [`
`  (7, "Peter"),`
`  (1, "Lilly"),`
`  (2, "Peter"),`
`  (12, "Peter"),`
`  (3, "Simon"),`
`  ],`
`  "3-Nice",`
`  "5-Naughty",`
`  "2-Nice",`
`  "1-Nice",`
`  ))`
	`Nice: Simon, Peter, Lilly`
	`Not found: Peter, Peter`

`print(naughty_or_nice_list(`
`  [`
`  (6, "John"),`
`  (4, "Karen"),`
`  (2, "Tim"),`
`  (1, "Merry"),`
`  (6, "Frank"),`
`  ],`
`  "6-Nice",`
`  "5-Naughty",`
`  "4-Nice",`
`  "3-Naughty",`
`  "2-Nice",`
`  "1-Naughty",`
`  Frank="Nice",`
`  Merry="Nice",`
`  John="Naughty",`
`))`
	`Nice: Karen, Tim, Frank`
	`Naughty: Merry, John`