# Problem 1. Thea the Photographer

Thea is a photographer. She takes pictures of people on special events. She is a good friend and you want to help her.

She wants to inform her clients when their pictures will be ready. Since the number of pictures is big and it requires time for editing (#nofilter, #allnatural) **every** single picture - you decide to write a program in order to help her.

Thea follows this pattern: first she takes **all** pictures. Then she goes through every single picture to **filter** these pictures that are considered "good". Then she needs to upload every single **filtered** picture to her cloud. She is considered ready when all **filtered** pictures are **uploaded** in her picture storage.

You will receive the **amount** of pictures she had taken. Then the approximate **time** in **seconds** for every picture to be filtered. Then a **filter factor** – a **percentage** (integer number) of the **total photos** (rounded to the nearest **bigger** **integer** value e.g. 5.01 -> 6) that are good enough to be given to her clients (Photoshop may do miracles but Thea does not). Approximate **time** for every picture to be uploaded will be given again in **seconds**. Your task is: based on this input to display total time needed for her to be ready with the pictures in given below format.

### Input

On the first line you will receive an integer **N** – the amount of pictures Thea had taken.

On the second line you will receive an integer **FT** – the amount of time (filter time) in seconds that Thea will require to filter every single picture.

On the third line you will receive an integer **FF** – the filter factor or the percentage of the total pictures that are considered “good” to be uploaded.

On the fourth line you will receive an integer **UT** – the amount of time needed for every **filtered** picture to be uploaded to her storage.

The input will be in the described format, there is no need to check it explicitly.

## Output

Print the amount of time Thea will need in order to have her pictures ready to be sent to her client in given format:

d:HH:mm:ss

d - days needed – starting from 0.

HH – hours needed – from 00 to 24.

mm – minutes needed – from 00 to 59.

ss – minutes needed – from 00 to 59.

## Constrains

The amount of total pictures Thea will have taken is range [0 … 1 000 000]

The seconds for both filtering and uploading will be in range [0 … 100 000]

The filter factor will be an integer number between [0 … 100].

## Examples

|  |  |  |
| --- | --- | --- |
| **Input** | **Output** | **Comments** |
| 1000  1  50  1 | 0:00:25:00 | Total pictures = 1 000, 50% of them are useful -> Filtered pictures = 500  Total pictures \* filter time = 1000 s  Filtered pictures \* upload time = 500 s  Total time = 1500 s |
| 5342  2  82  3 | 0:06:37:07 | Total pictures = 5342 - 82% of them are useful -> 4380.44-> 4381 filtered. |

# Problem 2. Trophon the Grumpy Cat

Trophon is very angry with his owner, because he left him alone during the teamwork defenses for the Software Technologies Course at SoftUni. It’s time for Trophon to get his payback and he will do it, by breaking various household items.

Each item has a **price rating** which is a number that describes how valuable is that item for Trophon’s owner. You will be **given an entry point** from which Trophon will start **breaking** the **items to his left**, and **then** to **his right**. Trophon **will never** break the item at the **entry point**.

You must calculate the damage to **both** his **left**, and **right**, then print **only the higher (bigger)** damage to the household. If both **sums** are **equal** print the **left-most** one.

## Input / Constrains

* On the **first line** you will receive the **price ratings**, separated by (space). Each element will be integer in range **[-231… 231]**
* On the **second line** you will receive the **entry point**, which will always be between the second and the penultimate element in the array
* On the **third line** you will receive the **type of items** Trophon wants to break, which will be one of the following:
  + **cheap** – items that have **lower price rating** than the entry point item
  + **expensive** – items that have **same price rating**, or **higher price rating** than the entry point item
* On the **last line** you will receive the **type of price ratings** that Trophon will look for, which will be one of the following:
  + **positive** – price ratings **above** 0
  + **negative** – price ratings **below** 0
  + **all** – **any** price ratings

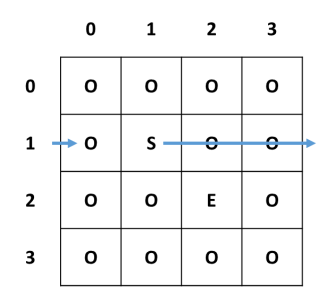
## Output

* Single line containing the sum of price ratings and their position based on the entry point in the following format:
  + “{**position**} – {**sum of price ratings**}”

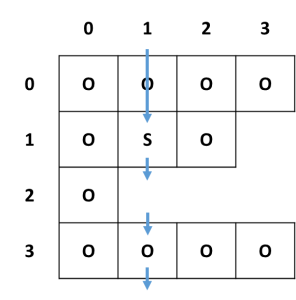
## Examples

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Output** |  | **Input** | **Output** |
| 1 5 1  1  cheap  all | Left - 1 |  | -2 2 1 5 9 3 2 -2 1 -1 -3 3  7  expensive  positive | Left – 22 |

# Problem 3. Portal

You are on the path to copyright a new technology that makes teleportation possible. As you are not sure if it is dangerous, the tests are conducted in a closed environment using robots. The room in which the tests are made is a **two-dimensional jagged array**.

The robot **always starts at the room start** (S) and his mission is to **escape** the room. He can travel in every one of the four directions by receiving input from the scientists. However, the robot **can teleport through the walls** so when he reaches the end of the room **he reappears from the opposite side**. For example, if a robot starts at [1, 1] in the shown matrix and he receives input "RRR", he will end up at [1, 0] as shown on the first picture.

If the room is not square and there is an empty space on the robot's path, he will simply teleport to the next existing cell in his direction. For example, if a robot starts at [1, 1] and he receives input "DDD", he will end up at [1, 1] as shown on the second picture.

Тhe room will have a **start** and **at least one exit**; all **other** cells are marked as empty:

* **Start (S)** – the cell from which the robot starts
* **Exit (E)** – the cell at which the robot exits the matrix
* **Empty cell** (O) – nothing (O is a character, not a number)

If a robot reaches the exit, print "Experiment successful. {turn count} jumps required."

If the robot can't find the exit print " Robot stuck at {row} {column}. Experiment failed."

## Input / Constrains

* On the **first line** of input you will get **N,** the **size of the matrix.** N will be in the range [0…100]
* On the next N lines, you will receive **the actual matrix**
* On the last line you receive a **string containing the directions the robot will be travelling**

## Output

* If therobot **can't reach the exit** print **"Robot stuck at R C. Experiment failed."** Where R is the row and C is the column that the robot died
* If the robot **reaches an exit** print **"Experiment successful. T turns required."** Where T is the number of turns that took the robot to reach the exit

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3  OOO  OSO  OOE  DR | Experiment successful. 2 turns required. |
| **Input** | **Output** |
| 5  OOOOS  OOOO  OO  OOOOO  OOOOE  DLUU | Robot stuck at 0 3. Experiment failed. |

# Problem 4. Files

You are given number of files with their **full file paths** and **file sizes**. You need to **print all file names** with **a given extension** that are present in **a given root directory sorted by their file size** in **descending order**. If two files have **same size**, order them **by alphabetical order**.

If a **file name (file name + extension) appears more than once in a given root**, **save only its latest value.** If a file name appears in more than one root, they are treated as different files.

If there **aren't any files** that correspond to the query, print **"No"**.

## Input / Constrains

* On the **first line** of input you will get **N** the **number of files to be read from the console**
* On the next N lines, you receive **the actual files in the format "root\folder\filename.extension;filesize"**
* There may be more than one folder e.g. **files can be deeply nested**
* On the last line you receive a query string in format **"{extension} in {root}"**. You need to print all files with the given extension that are in present in the given root

## Output

* You need to print all files sorted by their size in descending order.
* If two files have the same size, order them by alphabetical order.
* Files should be printed in the given format **"filename.extension - filesize KB"**
* If there **aren't any movies** that correspond to the query, print **"No"**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4  Windows\Temp\win.exe;5423  Games\Wow\wow.exe;1024  Games\Wow\patcher.cs;65212  Games\Pirates\Start\keygen.exe;1024  exe in Games | keygen.exe - 1024 KB  wow.exe - 1024 KB |
| 3  C:\Documents\01. problems.docx;6521  D:\Documents\02. Documents\ presentation.pptx;44234  E:\Movies\Classics\someclassicmovie.avi;6221235212  docx in E: | No |