**Problem 1 – Kompot**

It`s a summer. You can find fruits all around you. Mitko likes kompots very much, specially peaches, cherries and plums kompots and he wants to make some for the winter.

You may receive the following fruits:

* **Peach {0.200}**
* **Plum {0.120}**
* **Cherry {2.5}**

The number in the bracket is fruit weight in kilograms. You have to make kompots from this fruits as you know that 1 peach weight is 140 g, 1 plum weight is 20 g and 1 cherry weight is 9 g.

In the same time you know that for 1 kompot you must use the following numbers of fruits:

* **2.5 peach**
* **10 plum**
* **25 cherry**

After you make your kompots round them floor.

If you receive other kind of fruits collect them in a bucket. You will make rakiya from them. You will add some sugar and you will produce 0.200 l (liter) rakiya from 1 kilo fruit.

## Input

You will receive an **array of strings**. In the **first input element,** you will receive all kind of fruits and in the **second input element** you will receive the weight in kilograms.

## Output

* The output will be like the examples below. Rakiya must be formatted to the second number after the coma

## Constraints

* The **input will always be valid.**
* The **input will be a string containing only letters, numbers, spaces and comas**.
* Allowed working **time** / **memory**: **100ms** / **16MB**.

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [ 'cherry 1.2',  'peach 2.2',  'plum 5.2',  'peach 0.1',  'cherry 0.2',  'cherry 5.0',  'plum 10',  'cherry 20.0' ,  'papaya 20' ] | Cherry kompots: 117  Peach kompots: 6  Plum kompots: 76  Rakiya liters: 4.00 |
| [ 'apple 6',  'peach 25.158',  'strawberry 0.200',  'peach 0.1',  'banana 1.55',  'cherry 20.5',  'banana 16.8',  'grapes 205.65'  ,'watermelon 20.54'  ] | Cherry kompots: 91  Peach kompots: 72  Plum kompots: 0  Rakiya liters: 50.15 |

# Problem 2 – F1 Race

The race for the Grand Prix title of Sofia is about to start and all the pilots are in their positions! You are given an array of strings. The first string contains all the pilots in the race, separated by whitespace. The next strings contain actions with information about what happens in the race. Your task is to write a JS function that prints the final result of the race.

You may receive the following actions:

* Join {pilot}
* Crash {pilot}
* Pit {pilot}
* Overtake {pilot}

If you receive the **Join action**, you should **add** the pilot at last position in the race, but only **if** he isn`t in the race already.

If you receive the **Crash action**, the pilot crashes and you must **remove** him **if** he is in the race.

If you receive the **Pit action**, the pilot stops in the pit and you must **move** him one place down **if** he is in the race.

If you receive the **Overtake action**, the pilot overtakes the pilot in front of him and you must **move**  him one place up **if** he is in the race.

## Input / Consrtaints

You will receive an **array of strings**.

* In the **first input element,** you will receive all the pilots in the race – sequence of pilot names, separated by space.
* Each of the next elements will be an **action**.
* There will always be at least one pilot in the race.

## Output

* As output you must print all the pilots, separated by **“ ~ “**,at the end of the race.

## Constraints

* The **actions will always be valid.**
* The **pilot will be a string containing only letters from the alphabet**.
* Allowed working **time** / **memory**: **100ms** / **16MB**.

***Scroll down to see examples.***

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ["Vetel Hamilton Slavi",  "Pit Hamilton",  "Overtake Vetel",  "Crash Slavi"] | Vetel ~ Hamilton |
| ["Vetel Hamilton Raikonnen Botas Slavi",  "Pit Hamilton",  "Overtake LeClerc",  "Join Ricardo",  "Crash Botas",  "Overtake Ricardo",  "Overtake Ricardo",  "Overtake Ricardo",  "Crash Slavi"] | Vetel ~ Ricardo ~ Raikonnen ~ Hamilton |

# Problem 03 – DNAex

*Every living creature on Earth has genomes in it body. The genome is the total genetic material of an organism and includes both the genes and non-coding sequences. The genome size, and the number of genes it encodes varies widely between organisms (e.g. viruses and viroids have the smallest amount of genomes).*

The next task of candidate-intern Tanya is called „Genome“ and you have to help. You will receive different amount of **encrypted** genes each on a separate line. Until you receive a line “**Stop!**” you have to decrypt the information given and save some data that you will need later on. Each valid information about a gene should consist of:

* **Name of the gene** may contain some of the following characters between its letters (!@#$?). Example for valid names: “!@pro?#line!#” (proline)
* **Length of the name** with a “=” before it. (e.g. =12; =5…)
* **Count of genes** with a “--” before it. (e.g. --800; --142)
* **Organism that it belongs to with** “<<” before it. (e.g. <<cat; <<dog)

**Note: the info will be in that exact order. If something is missing the input is invalid and you should ignore it. If the length of the name does not match with the actual name given, the input is considered invalid and you should ignore it.**

Examples for **valid** input:

“!cad$$he!rins!@=9--30229<<human”

“!@leu?#cine!#=7--800<<cat”

Examples of **invalid** input:

“bx!=4--421<<bison” – the length 4 does not match with the actual length

“#nms!n--126<<dog” – the length is missing

**Store** the information about the genes and print all the organisms you have encountered with their **total** amount of genes.

## Input

* You will receive encrypted lines of input storing information about a gene until you receive “**Stop!**”
* all names will always be **lower-case** characters, **only Latin letters**
* the input will always be **valid** (no whitespaces)

## Output

* Print every organism with their total amount of genes (genome) in **descending order**
* For every organism print “**{organism name} has genome size of {genes count}**”
* If genes count equal -> **save the order** of the input

## Constraints

* all numbers will be in range [1 - 10000]
* name will be string [1 - 1000]

## Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| !@ab?si?di!a@=7--152<<human  b!etu?la@=6--321<<dog  !curtob@acter##ium$=14--230<<dog  !some@thin@g##=9<<human  Stop! | dog has genome size of 551  human has genome size of 152 |
| =12<<cat  !vi@rus?=2--142  ?!cur##viba@cter!!=11--800<<cat  !fre?esia#=7--450<<mouse  @pa!rcuba@cteria$=13--351<<mouse  !richel#ia??=8--900<<human  Stop! | human has genome size of 900  mouse has genome size of 801  cat has genome size of 800 |
| !@ру?би#=4--57<<polecat  ?do?@#ri#=4--89<<polecat  =12<<cat  !vi@rus?=2--142  @pa!rcu>ba@cteria$=13--234<<mouse  ?!cur##viba@cter!!=11--680<<cat  Stop! | cat has genome size of 680  polecat has genome size of 89 |

# Problem 4 – F1 Championship

Write a JavaScript program that **calculate and display** F1 teams and pilots points in specific format.

As **input** you will receive an array of strings.

Each string will consist of the following information with format:

“Team name -> Pilot name -> Pilot points”

Team name will always be valid string, Pilot name will also be valid string, Pilot points will be a number. Your task is to print the result like the example below. Every pilot can take points from more than one championship. Every team will have only two pilots, no more, no less. Every pilot will be only in one team.

### Input

You will receive **one argument –** an **array strings** as shown above.

### Output

Print on the **console** the **top 3 teams** with biggest sum of **pilot points** as the **first criteria** and after that sort the **pilots in every team** depending on their points.

### Constraints

* The **number** of **points** in the **input argument** will be valid positive number
* There **will** be **no invalid** **input**
* There **will** be **no teams or pilots with equal points**

### Examples

|  |
| --- |
| **Input** |
| ["Ferrari -> Kimi Raikonnen -> 25",  "Ferrari -> Sebastian Vettel -> 18",  "Mercedes -> Lewis Hamilton -> 10",  "Mercedes -> Valteri Bottas -> 8",  "Red Bull -> Max Verstapen -> 6",  "Red Bull -> Daniel Ricciardo -> 4"] |
| **Output** |
| Ferrari: 43  -- Kimi Raikonnen -> 25  -- Sebastian Vettel -> 18  Mercedes: 18  -- Lewis Hamilton -> 10  -- Valteri Bottas -> 8  Red Bull: 10  -- Max Verstapen -> 6  -- Daniel Ricciardo -> 4 |