# Exercises: Regular Expressions (RegEx)

This document defines the **exercise assignments** for the ["Programming Fundamentals Extended" course @ Software University](https://softuni.bg/courses/programming-fundamentals). Please submit your solutions (source code) of all below described problems in [Judge](https://judge.softuni.bg/Contests/444).

## Cards

You will be given sequences of playing cards. Your task is to print the valid ones of them, separated by a **comma** and a **space**.

The **VALID playing cards** have:

* **power** – **2**, **3**, **4**… **10**, **J**, **Q**, **K**, **A**.
* **suit** – **S**, **H**, **D**, **C**

A card is formed by its power and its suit in the following format: {power}{suit}. . ., but **ONLY** those with **valid** **power** and **valid suit**, should be **considered** **VALID**.

So a valid car would look like this: **KS**, **10S**, **2D**, **3D**.

### Input

The input will consist of a **single line**, containing a sequence of cards.

### Output

The output should be a single line, containing the valid cards, separated by a **comma** and a **space**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2S3S4S5S6S | 2S, 3S, 4S, 5S, 6S |
| 2SASKS6SJSQSOS | 2S, AS, KS, 6S, JS, QS |

## Word Encounter

You will be given a **filter**, in the form of a **string** of **two characters** – the **first** being an **ASCII character**, and the **second** – a **digit**.

You will then receive a **sequence of sentences**. You must extract **all words** from those sentences, and print **only** the **words** that **contain** the **given filter character**, **at least** **N times** – **N** being the **filter digit**.

There are **2 types** of sentences – **Valid** and **Invalid**. The **valid ones**, always **start** with a **capital** **letter**, and **always** **end** with one of the following characters: “**.**”, “**!**”, “**?**”. Invalid sentences, should be ignored.

**Note**: You **WILL NOT** be given more than **1 sentence** on a **single input line**.

The input sequence ends, when you receive the command “**end**”. After that you must print all the filtered words, you’ve gathered.

The **valid words** must be printed, on a **single line**, **separated** by a **comma** and a **space**.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| l2  This will, likely be a funny feeling, Laslo.  Will you come to my playlife ;)?  end | will, likely, Will, playlife |
| o1  How about... No...  Maaan, this is amazing! Yeah; I know bro!  end | How, about, No, know, bro |

### Hint

Check if there is a way to find where a word, in a sentence, starts, and ends. There surely must be a way to do that with Regular Expressions.

## Booming Cannon

You just bought brand new cannon, but you are not very good at calculating where your bullets will fall. Instead of just shooting blindly you decided to make a simulation, which is expensive and a bit dangerous, you decided to simulate cannon shots using strings.

On the **first** line, you will receive an **array** of **integers**. **First** element – **n** will be the **distance** of the **shot** and the **second** element – **m** will be the **count** of **elements**, which the shot will **destroy**.

On the **second** line, you will receive a **string**. Every cannon in the string will be marked with "\**<<<**". You have to skip next **n** characters after the cannon and destroy the next **m** characters.

If you find **another** cannon in your **target** 🡺 take as target only until the **last** **character** **before** the new cannon and start **new** **shot** with the **new** **cannon**.

Your task is to print all destroyed targets separated with "**/\**".

### Output

Print **all** destroyed **targets** separated with "**/\** ".

### Examples:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 8  Village\<<<FieldSoldiersForest\<<<FieldTwoTanksNothing | Soldiers/\TwoTanks |
| 0 4  <<invalid\<<<miss|noMoreTagets | miss |

## Grocery Shop

You are given the task to write a program for a shop, which calculates the total price of products. The problem is the old system was full of bugs and you need to decrypt the data which you receive.

We know the **following** about the **input** data:

* The product name **starts** with **English capital** letter and **all** **other** letters should be **lower-case**.
* After that the product name **should** be **followed** by **colon** – ‘**:**’
* Price of the **product**, which should be **floating** **point** number with exactly **two** digits **after** the **decimal point**.

If you receive input, which does **not** follow the rules above 🡺 **ignore** it. If you receive **new** **price** for already **existing** **product** 🡺 **rewrite** the previous values.

At the end **print** the **prices** for **every** product. **Order** the **products** by price in descending order.

### Input

You will receive indefinite count of string until you receive the command "**bill**".

### Output

Print **all** prices ordered by **price** in **descending** order. Use the following **format**:

{nameOfTheProduct} costs {priceOfTheProduct}

**Format** the price to the **2nd decimal place**.

### Constraints

* The average prices will be in the interval **[0.01…50.00]**

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| Bread:12.31  bRead:41.31  Cola-12.31  Cola:12.31  Cola:14.23  bill | Cola costs 14.23  Bread costs 12.31 |

|  |  |
| --- | --- |
| **Input** | **Output** |
| Water:12.31  Water:12,42  Water:15.31  WaTer:30.20  Beer:15.12  bill | Water costs 15.31  Beer costs 15.12 |

## SoftUni Messages

After one of the beer fests at SoftUni (called also Friday by the people outside SoftUni) Vladi made a secret messaging system, which will help the SoftUni team to safely communicate. The problem is he lost the code which decrypted the messages and now he needs you to write it again.

All he knows is the **pattern**, which he used to decrypt the messages. Valid messages should be in the following **format**:

* Have length of **m**, where **m** is the **number**, which is given **after** **every** input line
* Have **one or more** **digits** **before** the message
* Consist **only** of **Latin** alphabet **letters**
* Have at least **one** **number** **after** the message
* Have **no** **letters** **after** the **message**

After you find the message you need to **decrypt** it. Take **all** number **before** and **after** the message and **consider** them as **indices**. If they **exist** in the message form a **string** from the **characters** at those **indices** in the **message**. If they do **not** exist – just **ignore** them.

### Input

* On every **odd** **line**, you will receive an **encrypted** message
* On every **even** **line**, you will receive the **searched** **length** for the message from the **previous** line.
* When the line “**Decrypt!**” is entered, the input sequence **ends**.

### Output

* For every **valid** message **print**:

{message} = {verificationCode}

### Constraints

* The input lines can consist of **ANY ASCII** character.
* There will be **NO** such cases as an encrypted message without a number before it.
* The number will be a valid integer in the range [0, 100].
* You will receive only positive numbers as indices in the string.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| 201masrt347  5  201masrt34  2  2reB]\1;10  3  Decrypt! | masrt = smart  reB = Beer |

|  |  |
| --- | --- |
| **Input** | **Output** |
| 1owpce000001  5  Perde3124  5  12lol31]1\3a  3  Decrypt! | owpce = wooooow |

## Fish Statistics

You are a marine biologist tasked with researching various types of fish. You will receive a **single line** on the console as input. From this line, you must extract **all the fish** you find and **print statistics** about **each** **one**.  
Fish are categorized by three criteria: **tail length**, **body length** and **status**. A standard fish looks like this:

|  |
| --- |
| ><(((('> |

This fish has a **tail length** of **1**, a **body length** of **4** and has the **status** “**Awake**”, since its **eye is open**. **One ASCII character** represents 2 **centimeters** in real life. By those standards, this fish has a **tail length** of **2 cm** and a **body length** of **8 cm**. There are various **types** of **tails**, **bodies** and **statuses**, which are described below:

* Tail types:
  + Tail **longer** than **5** “<” characters 🡪 Long
  + Tail **longer** than **1** “<” characters 🡪 Medium
  + Tail, which is **1** “<” character long 🡪 **Short**
  + Nonexistent tail 🡪 **None**
* Body types:
  + Body **longer** than **10** “(” characters 🡪 Long
  + Body **longer** than **5** “(” characters 🡪 Medium
  + Any other length 🡪 **Short**
* Statuses:
  + **'** 🡪 Awake
  + **-** 🡪 Asleep
  + **x** 🡪 Dead

The input will contain a **variable amount of fish**, separated by any sequence of **ASCII characters**. There’s a **possibility** you might receive input, which has **no fish** – in this case, just print “No fish found.”, and end the program.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| ><(((('> >>>><((((((((('>~~~~~<((->~~~ o o >>>><((x> | Fish 1: ><(((('>  Tail type: Short (2 cm)  Body type: Short (8 cm)  Status: Awake  Fish 2: >>>><((((((((('>  Tail type: Medium (8 cm)  Body type: Medium (18 cm)  Status: Awake  Fish 3: <((->  Tail type: None  Body type: Short (4 cm)  Status: Asleep  Fish 4: >>>><((x>  Tail type: Medium (8 cm)  Body type: Short (4 cm)  Status: Dead |
| o oo >>>><((-> \* ()()()(): | Fish 1: >>>><((->  Tail type: Medium (8 cm)  Body type: Short (4 cm)  Status: Asleep |
| o o ><(-> >><(('> <(((((((((x> \* #%#$@ \* | Fish 1: ><(->  Tail type: Short (2 cm)  Body type: Short (2 cm)  Status: Asleep  Fish 2: >><(('>  Tail type: Medium (4 cm)  Body type: Short (4 cm)  Status: Awake  Fish 3: <(((((((((x>  Tail type: None  Body type: Medium (18 cm)  Status: Dead |
| o xx xxxx ~ ~ ~ xxxx | No fish found. |

## Happiness Index

You will be given a **string**, consisting of one or several **emoticons** and random **garbage characters** in-between them. Your task is to count the **happy** and **sad** emoticons and calculate the **happiness index** of the string. The index is calculated by this formula: {happyEmoticonsCount} / {sadEmoticonsCount}. The happiness index is then **rounded** to the **second** decimal place. **Two** emoticons will **never touch**.

The emoticons of all the emotion types are as follows:

* Happy: :), :D, ;), :\*, :], ;], :}, ;}, (:, \*:, c:, [:, [;
* Sad: :(, D:, ;(, :[, ;[, :{, ;{, ):, :c, ]:, ];

After you calculate the happiness index, **print** the final emoticon score, following this format:

* Happiness index **greater than or equal** to **2** 🡪 :D
* Happiness index **greater than 1** 🡪 :)
* Happiness index **equal** to **1** 🡪 :|
* Happiness index **smaller than 1** 🡪 :(

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| :)^%&:)\*\*&:]v;)ff:( | Happiness index: 4.00 :D [Happy count: 4, Sad count: 1] |
| &&:(&:)z:)zz%%!%%!%%!%:( | Happiness index: 1.00 :|  [Happy count: 2, Sad count: 2] |
| <<<<<<<<<<<]:bb:[<<<<<<<<<:}<<<<<<:(<<<<<<:)z:)z:)&@^@%@ | Happiness index: 1.33 :) [Happy count: 4, Sad count: 3] |
| ;(&m%td[:^i+@#:{eqk#n | Happiness index: 0.50 :(  [Happy count: 1, Sad count: 2] |

### Hints

* Regex [named capturing groups](http://www.regular-expressions.info/named.html) might come in handy for this problem.

## \* Commits

Believe it or not, GitHub doesn’t process its commits using software. They achieve their versatile version control by employing the services of a small man, named Master Branch. Mr. Branch is sick of parsing the huge amounts of data every single day and needs your help.

Write a program, which receives **lines** of **URLs**. Until you receive the command “git push”, your task is to **parse** the URLs, which come in the following format:

**https://github.com/{user}/{repo}/commit/{hash},{message},{additions},{deletions}**

Valid input will follow this format. If any input doesn’t follow this format, **ignore it**. The validation rules are as follows:

* The **username** may contain **alphanumeric characters** (Latin letters and digits) and **hyphens** (-).
* The **repository name** may contain **letters**, **hyphens** and **underscores**.
* The **commit hash** will be a **SHA-1** hash, and as such, it may contain **only** **hexadecimal** **characters** (digits and letters A through F). Since SHA-1 hashes are always 20 bytes long, the commit hash will also **always** be **40 characters** long.
* The message may contain **any** character, **except** the **new** **line** character.
* The **additions** and **deletions** are numbers and may **only** contain **digits**.

While parsing the input, you also need to **categorize** it for Mr. Branch. Every **user** has his own **repos** (a user cannot have two repos with the **same name**) and every **repo** has a list of **commits**. The users are **sorted alphabetically by name**. The repos are also **sorted alphabetically by name**. The **commits** are sorted by **order of insertion**.

After you receive the command “git push”, your task is to print information about the commits in the following format:

|  |
| --- |
| {user}:  {repo}:  commit {hash}: {message} ({additionsCount} additions, {deletionsCount} deletions)  commit {hash}: {message} ({additionsCount} additions, {deletionsCount} deletions)  …  Total: {totalAdditionsCount} additions, {totalDeletionsCount} deletions |

### Examples

|  |
| --- |
| **Input** |
| https://github.com/gosho/http-parser/commit/f17c563aed112dabbdbe977fcdb88772be7d85eb,general fixes,14,3 https://github.com/pesho-1232/db-checker/commit/5ca49ccc157c98af2c71391223e4b254ee327134,fix SELECT statement,9,19 https://github.com/gosho/http-parser/commit/1f0abbdf5006b4a88aed1b72f9a937b35a5126dc,One does not simply merge into master,1,15 https://github.com/stamat4o/hackertools/commit/ddb473ab0304e5e843983da8b26925dbb3495afa,another big bag of changes,8,18 **git push** |

|  |
| --- |
| **Output** |
| gosho:  http-parser:  commit f17c563aed112dabbdbe977fcdb88772be7d85eb: general fixes (14 additions, 3 deletions)  commit 1f0abbdf5006b4a88aed1b72f9a937b35a5126dc: One does not simply merge into master (1 additions, 15 deletions)  Total: 15 additions, 18 deletions  pesho-1232:  db-checker:  commit 5ca49ccc157c98af2c71391223e4b254ee327134: fix SELECT statement (9 additions, 19 deletions)  Total: 9 additions, 19 deletions  stamat4o:  hackertools:  commit ddb473ab0304e5e843983da8b26925dbb3495afa: another big bag of changes (8 additions, 18 deletions)  Total: 8 additions, 18 deletions |

|  |
| --- |
| **Input** |
| https://github.com/A4B5TR/ceca-music-player/commit/5ca49ccc157c98af2c71391223e4b254ee327134,implement mile kitic,16,16  https://github.com/Huosey/3ds-hax/commit/c4994a26e7370d9e482e9317c9a0489648c83fc6,for free,8,20  https://github.com/A4B5TR/ceca-music-player/commit/136aa8bd1ac90d58230767027db5d2d0f3a6b9a5,lots and lots of changes,12,3  https://github.com/A4B5TR/ceca-music-player/commit/d5cdb78e9a10af7f929dfa070577ef548bdadbb9,stuff,17,3  https://github.com/dirtyhaxxor/csbotrepo/commit/8d99397247811cdc0210a92c9beb21bb20689dbc,add rush b functionality,1,6  **git push** |

|  |
| --- |
| **Output** |
| A4B5TR:  ceca-music-player:  commit 5ca49ccc157c98af2c71391223e4b254ee327134: implement mile kitic (16 additions, 16 deletions)  commit 136aa8bd1ac90d58230767027db5d2d0f3a6b9a5: lots and lots of changes (12 additions, 3 deletions)  commit d5cdb78e9a10af7f929dfa070577ef548bdadbb9: stuff (17 additions, 3 deletions)  Total: 45 additions, 22 deletions  dirtyhaxxor:  csbotrepo:  commit 8d99397247811cdc0210a92c9beb21bb20689dbc: add rush b functionality (1 additions, 6 deletions)  Total: 1 additions, 6 deletions |

|  |
| --- |
| **Input** |
| https://github.com/prakash/thymeleef/commit/e97c8637a13bf911e55030681884c8301a67e1dd,[MAJOR FEAT] minor fix,8,7  https://github.com/ivanov33/superstartup/commit/56e566a92b48e430b4a81fd05fe777fbd612e085,it's 5am. It works. I quit.,9,2  https://github.com/ivanov34/superstartup/commit/460afc15d7f7ab14be7966bc13019c17de96b3b2,sorry I'm new. merged every branch to master,1990048,2913460  https://github.com/prakash/thymeleef/commit/f17c563aed112dabbdbe977fcdb88772be7d85eb,[MINOR FEAT] fixed spelling error,19,14  https://github.com/ivanov35/superstartup/commit/ecd3e802dec29c41e7f4d653e1019749f4ca6eec,ivanov34 got fired.. rebased to 3rd commit..,6,20  https://github.com/ivanov34/superstartup/commit/71c2c02ccf8da0765d21a79bf9bcfe1dd87f1544,im back,18,2  https://github.com/adamash/thyme$leef/commit/58033134ca5bfb1c2cd606513f02b854ba7529c9,hehe,1,2  https://github.com/prakash/thymeleef/commit/44b801d6aa6e37d5960d14734d10e87cfc6ec0a8,I'm done,16,0  git push |

|  |
| --- |
| **Output** |
| ivanov33:  superstartup:  commit 56e566a92b48e430b4a81fd05fe777fbd612e085: it's 5am. It works. I quit. (9 additions, 2 deletions)  Total: 9 additions, 2 deletions  ivanov34:  superstartup:  commit 460afc15d7f7ab14be7966bc13019c17de96b3b2: sorry I'm new. merged every branch to master (1990048 additions, 2913460 deletions)  commit 71c2c02ccf8da0765d21a79bf9bcfe1dd87f1544: im back (18 additions, 2 deletions)  Total: 1990066 additions, 2913462 deletions  ivanov35:  superstartup:  commit ecd3e802dec29c41e7f4d653e1019749f4ca6eec: ivanov34 got fired.. rebased to 3rd commit.. (6 additions, 20 deletions)  Total: 6 additions, 20 deletions  prakash:  thymeleef:  commit e97c8637a13bf911e55030681884c8301a67e1dd: [MAJOR FEAT] minor fix (8 additions, 7 deletions)  commit f17c563aed112dabbdbe977fcdb88772be7d85eb: [MINOR FEAT] fixed spelling error (19 additions, 14 deletions)  commit 44b801d6aa6e37d5960d14734d10e87cfc6ec0a8: I'm done (16 additions, 0 deletions)  Total: 43 additions, 21 deletions |

### Hints

* You can store information about the commits in a custom **Commit** class, which contains the **commit hash**, **message**, **additions** and **deletions**. After that, calculating the total additions and deletions per repo is only a **LINQ** query away.