# Problem 1 – Username Parser

Every online platform should have username validation. Your task is to write initial username parser and list all the valid usernames with the given condition.

Write a PHP script that takes as input a **list of strings** and displays them as **HTML list** where the lines longer than certain length are displayed normally, while others are **removed** or **colored red**, based on **the option** given.

For **example**, if the **text** is "**Aha**", the **length** is **5** and the **show** checkbox is **selected**, the **text** is displayed with r**ed color**. If the **show** checkbox is **not selected**, the **text** is **not** displayed at all.

The output should be a HTML list **<ul><li>…</li></ul>**. Ensure you encode correctly in the output all HTML special characters like **&**, **<**, **>** and **"** (use **htmlspecialchars()** function in PHP).

### Input

The input will be read from an **HTTP GET** **request** holding parameters named **list**, **length** and **show**, specifying the list of strings, the minimum length limit and whether to show or not the usernames, shorter than the limit. The input strings are separated one from another by a new line. All whitespace at the start and at the end of each line is ignored, as well as lines holding only whitespace.

### Output

The output is the **HTML list** holding the input strings, where longer strings are abbreviated. See the example below.

### Constraints

* The **list** is a text area holding strings, each at separate line, eventually with whitespace at its ends. The list will hold only **ASCII** characters (no Unicode).
* The **length** is a text field holding an integer number in the range [1…1000].
* The **show** is a checkbox field, which can be either selected or not.
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Example

|  |  |
| --- | --- |
| **Input** | |
| list | Angel  Ivancho  Aha  Toni  Pesho  Gosho |
| length | 5 |
| show | on |
| **Output** | |
| <ul><li>Angel</li><li>Ivancho</li><li style="color: red;">Aha</li><li style="color: red;">Toni</li><li>Pesho</li><li>Gosho</li></ul> | |
| **Output in the Browser** | |
|  | |

# Problem 2 – Sum of All Values

You are given a **keys string** and a **text string**. Write a PHP script that finds the **start key** and the **end key** from the **keys string** and then **applies** **them** to the **text string**.

The **start key** will **always** stay in the **beginning** of the **keys string**. It can contain **only letters and underscore** and **ends** just before the **first found digit**.

The **end key** will **always** stay in the **end** of the **keys string**. It can contain **only letters and underscore** and **starts** just after the **last found digit**.

Print at the console the **sum of all values** in the **text string,** between a **start** **key** and an **end key**. If the value is 0 then print “The total value is: *nothing*”. If no start key or no end key is found then print “A key is missing”.

### Input

The input will be read from an **HTTP GET** **request** holding parameters named **keysString and textString**.

### Output

The output should hold the **result text**, printed in HTML paragraph. The actual value of the sum should be ***italic.***

### Constraints

* The **keys string and text string** will hold only **ASCII** characters (no Unicode).
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12adghfgh243212gadghfgs43**endKEY** |
| textString | asdjykulgfjddfsffd**startKEY***12***endKEY**qwfrhtyu67543rewghy3tefdgd **startKEY***123.45***endKEY**wret34yre**startKEY***2.6***endKEY**213434ytuhrgerweasfd **startKEYendKEYstartKEY***asfdge***endKEY** |
| **Output** | |
| <p>The total value is: <em>138.05</em></p> | |

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12**a** |
| textString | asdjykulgfjddfsffd**startKEY***12endKEYqwfrhtyu67543rewghy3tefdgdst***a**rtKEY123.45endKEYwret34yre**startKEY***2.6endKEY213434ytuhrgerwe***a**sfd **startKEY***endKEYst***a**rtKEYasfdgeendKEY |
| **Output** | |
| <p>The total value is: <em>nothing</em></p> | |

|  |  |
| --- | --- |
| **Input** | |
| keysString | **startKEY**12 |
| textString | asdjykulgfjddfsffdstartKEY12endKEYqwfrhtyu67543rewghy3tefdgd |
| **Output** | |
| <p>A key is missing</p>; | |

# Problem 3 – Affine Cipher

Write a PHP script to **encrypt a matrix** of strings **using affine cipher** on the **capital letters** of the **Latin alphabet**. The formulae of the cipher is **E(x) = (k\*x + s) % m** where **m** is the size of the current alphabet, **k** and **s** are the first and second keys of the cipher, **x** is the position of the letter (starting from 0) in the alphabet. Special characters should not be ciphered. You should print out a **HTML table** holding the ciphered text.

### Input

The input comes as **JSON-encoded matrix** and will be read from an **HTTP GET** **request** holding a parameter named **jsonTable**. The first row of the matrix holds the text to be ciphered as **array of strings**. The second row of the matrix holds an **array of integers** which hold the values of **the two keys**.The input data will always be **valid** and in the format described. There is no need to check it explicitly.

### Output

The output should be a **HTML table** that shows the ciphered text in the matrix (each word on a separate row, each letter in separate cell), colored by changing the cells' background to **#CCC** (see the examples below) where the cells of the matrix are not empty. Ensure all your cell data is correctly encoded as HTML. **Follow strictly the sample HTML output format below.**

### Constraints

* The input **JSON string** will always contain an array of arrays holding an array of strings and array of integers having.
* The key **k** will be in the range [1…9].
* The key **s** will be in the range [0…9].
* The minimal **size** of the html table will be 1 x 1.
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output - Visualized** |
| [["god","save","the","queen"],[7,2]] |  |
| **Output** | |
| <table border='1' cellpadding='5'><tr><td style='background:#CCC'>S</td><td style='background:#CCC'>W</td><td style='background:#CCC'>X</td><td></td><td></td></tr><tr><td style='background:#CCC'>Y</td><td style='background:#CCC'>C</td><td style='background:#CCC'>T</td><td style='background:#CCC'>E</td><td></td></tr><tr><td style='background:#CCC'>F</td><td style='background:#CCC'>Z</td><td style='background:#CCC'>E</td><td></td><td></td></tr><tr><td style='background:#CCC'>K</td><td style='background:#CCC'>M</td><td style='background:#CCC'>E</td><td style='background:#CCC'>E</td><td style='background:#CCC'>P</td></tr></table> | |

# Problem 4 – Book Store

Write a PHP program that reads several **lines** of information about **books**, **filters the books in a given price range** and prints them in HTML **<div></div>** tags.

You will be given a **list**, **minimum price**, **maximum price**, **sort** and **order**. You must print **only the products in the price range between minimum and maximum price**. These products should be **sorted by genre, author or publish date** in **ascending**/**descending** order. If there **are equal genres or authors**, a **secondary sorting** should be performed by **publish date** in ascending order.

### Input

The input data should be read from an **HTTP GET request**. The product list will come from the **textarea 'text'**, holding each product on a separate line. The minimum and maximum price will come from input fields, respectively **'min-price'** and **'max-price'**. The sorting criteria will come from the input field **'sort'**. The order will come from the input field **'order'**.

### Output

Print each book with a price in the given range (inclusive). Each book should be printed in its own **<div></div>** tags. The book’s name should be printed in **<p></p>** tags. The other data should be printed as list items **<li></li>** in an unordered list **<ul></ul>**. The output should be exactly like in the example below. Ensure your output is properly encoded (use the **htmlspecialchars()** function).

### Constraints

* The books will be in the format **[author] /[name] /[genre] /[publish date] /[info]**.
* **Book prices** will be valid numbers in the range [0…100 000]. The symbol '**.**' will be used as decimal point. **Minimum price** will always be less than or equal to the **Maximum price.**
* The sort criteria will be one of the following: **‘genre’, ‘author’, ‘publish date’.**
* The sorting can be either **ascending** or **descending**.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | | |
| minPrice | | 5.00 |  | sort | author |
| maxPrice | | 10.45 |  | order | ascending |
| text | Gambardella, Matthew/XML Developer's Guide/Computer/44.95/2000-10-01/An in-depth look at creating applications with XML.  Ralls, Kim/Midnight Rain/Fantasy/19.15/2000-12-16/A former architect battles corporate zombies, an evil sorceress, and her own childhood to become queen of the world.  Corets, Eva/Maeve Ascendant/Fantasy/6.95/2000-11-17/After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.  Corets, Eva/Oberon's Legacy/Fantasy/5.00/2001-03-10/In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.  Randall, Cynthia/Lover Birds/Romance/5.95/2000-09-02/When Carla meets Paul at an ornithology conference, tempers fly as feathers get ruffled.  Thurman, Paula/Splish Splash/Romance/4.95/2000-11-02/A deep sea diver finds true love twenty thousand leagues beneath the sea. | | | | |

|  |
| --- |
| **Output** |
| <div><p>Maeve Ascendant</p><ul><li>Corets, Eva</li><li>Fantasy</li><li>6.95</li><li>2000-11-17</li><li>After the collapse of a nanotechnology society in England, the young survivors lay the foundation for a new society.</li></ul></div><div><p>Oberon's Legacy</p><ul><li>Corets, Eva</li><li>Fantasy</li><li>5.00</li><li>2001-03-10</li><li>In post-apocalypse England, the mysterious agent known only as Oberon helps to create a new life for the inhabitants of London. Sequel to Maeve Ascendant.</li></ul></div><div><p>Lover Birds</p><ul><li>Randall, Cynthia</li><li>Romance</li><li>5.95</li><li>2000-09-02</li><li>When Carla meets Paul at an ornithology conference, tempers fly as feathers get ruffled.</li></ul></div> |
|  |
| **Output (formatted)** |
|  |