# Problem 1 – Santa

Santa is modern nowadays. He only reads **children's wishes** if they are in a **special template**. He uses a predefined template and fills it just before dispatching the gift to every child – if the child was good, it gets the wanted present. If not – it gets a riddle to solve until the next Christmas.

Write a PHP script that takes as input a **child name**, a **wanted present** and **riddles** and puts them in the **Santa's template**.

The template is:

"$giftOf**[child name]** = $[wasChildGood] ? '**[wanted present]**' : '**[picked** **riddle]**';"

The filled values of should represent the current input.

The **child name** should has its whitespaces replaced by a **dash (‘-‘)**.

The **picked riddle** is the **n-th** riddle from the input, where **n** is the **length of the child name.** If the child's name has **more letters** than the **count of the riddles**, you continue counting the riddles from the first one.

For example "Petyrcho" has 8 letters. There are 3 riddles. You take the 2nd riddle (8 % 3 = 2).

### Input

The input will be read from an **HTTP GET** **request** holding parameters named **childName, wantedPresent** and **riddles**. The input "**riddles"** holds many riddles, separated by a **semicolon (';')**.

### Output

The output should hold the **filled template** **to Santa** on a **single line**. Ensure you handle correctly all HTML special chars with the **htmlspecialchars()** PHP function when you print the output.

### Constraints

* The **childName** and **wandtedPresent** are text fields holding non-empty strings. They may hold more than one word. There won't be any trailing whitespaces.
* The **riddles** is a text area, holding one single text line.
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Example

|  |  |
| --- | --- |
| **Input** | |
| childName | Petyrcho |
| wantedPresent | Big Truck |
| riddles | What do you call bears with no ears?;What is black and white and red all over?;Why do bears have fur coats? |
| **Output** | |
| $giftOfPetyrcho = $[wasChildGood] ? 'Big Truck' : 'What do you call bears with no ears?'; | |

# Problem 2 – Future Dates

You are given a **numbers string** and a **date string**. Write a PHP script that finds all the numbers in **numbers string** that are surrounded by **any character other than a letter**. It then **sums** the numbers and stores the sum **backwards.** If the sum is **58,** the **backwards sum** will be **85.**

The date string contains dates in the format **[YYYY-MM-DD]**, where **YYYY** is the year, represented with 4 digits, **MM** and **DD** are the month and day respectively, represented with 2 digits, with 0 if needed.

Print the **list of all dates**, found in the date string, **increased** by the **backwards sum**. Put the new dates as separate **list items (<li></li>)** in **unordered list (<ul></ul>)** in the same sequence as they appear.

If there are **no dates** in the date string, print “**No dates**” in a single **paragraph (<p></p>)**.

### Input

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

### Output

The output should hold the **resulted unordered list**, or a single **paragraph,** holding“**No dates**”.

### Constraints

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

### Examples

|  |  |
| --- | --- |
| **Input** | |
| numbersString | Th1s **12**# is \_**43**$ just %**2**^ random5text!!**1**! |
| dateString | 2014-12-22, this is today! Good luck with the exam. Yesterday was 21/12/2014. Three years ago was Friday 22-12-2011 and it was also working day, but 2011-12-24 was not! |
| **Output** | |
| <ul><li>2015-03-17</li><li>2012-03-18</li></ul> | |
| **Explanation** | |
| Sum in numbersString is 12 + 43 + 2 + 1 = 58. Backwards sum is 85.  2014-12-22 + 85 days = 2015-03-17  2011-12-24 + 85 days = 2012-03-18 | |

|  |  |
| --- | --- |
| **Input** | |
| numbersString | Ha-Ha-Ha-1. |
| dateString | Good luck with the exam! |
| **Output** | |
| <p>No dates</p> | |
| **Explanation** | |
| Sum in numbersString is 1. Backwards sum is 1.  No dates | |

# Problem 3 – Message Decoder

You are givenan **array of inputs** with two rows. The first row of the input contains a **number**. The second row contains an **array of strings** with ping transmission information. Your task is to write a PHP script that decodes a secret message hidden in between the ping information. The ping replies are in the format:

“Reply from [***ip\_address***]: bytes=[***size\_in\_bytes***] time=[***time\_in\_milliseconds***]ms TTL=[***time\_to\_live***]”

e.g. “Reply from **95.101.195.91**: bytes=**32** time=**068**ms TTL=**49**"

To find the secret message you should extract the **milliseconds** in the **reply time** and find the **Latin letters** and **spaces** corresponding to the milliseconds. After revealing the message, you should print out a HTML table holding the message. The table’s **columns** are equal to the number in the **first row** of the **input**. If a word goes **outside** the table’s boundary you should continue on a **new row** completing the **unfinished** row with empty cells. Every **separate** **word** should be **printed** on a **new row** as well. **Whitespace** is considered as empty (<td></td>) table cell and the symbol '\*' as **end** of a word.

### 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 input holds the **number of columns** in the matrix. The second row of the matrix holds an **array of strings** which hold the pinging information. There **won’t** be more than **one** space.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 hidden message (each word on a separate row, each letter in separate cell), colored by changing the cells' background to **#CAF** (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 integer value and an array of strings.
* The **column size** will be in the range [1…9].
* Allowed working time: 0.2 seconds. Allowed memory: 16 MB.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output - Visualized** |
| [4,["Ping results:",  "Reply from 95.101.195.91: bytes=32 time=115ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=111ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=102ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=116ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=117ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=110ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=105ms TTL=49"]] |  |
| **Output** | |
| <table border='1' cellpadding='5'><tr><td style='background:#CAF'>s</td><td style='background:#CAF'>o</td><td style='background:#CAF'>f</td><td style='background:#CAF'>t</td></tr><tr><td style='background:#CAF'>u</td><td style='background:#CAF'>n</td><td style='background:#CAF'>i</td><td></td></tr></table> | |
| **Input** | **Output - Visualized** |
| [5,["Ping results:",  "Reply from 95.101.195.91: bytes=32 time=083ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=111ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=102ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=116ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=119ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=097ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=114ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=101ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=042ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=085ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=110ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=105ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=118ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=101ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=114ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=115ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=105ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=116ms TTL=49",  "Reply from 95.101.195.91: bytes=32 time=121ms TTL=49"]] |  |
| **Output** | |
| <table border='1' cellpadding='5'><tr><td style='background:#CAF'>S</td><td style='background:#CAF'>o</td><td style='background:#CAF'>f</td><td style='background:#CAF'>t</td><td style='background:#CAF'>w</td></tr><tr><td style='background:#CAF'>a</td><td style='background:#CAF'>r</td><td style='background:#CAF'>e</td><td></td><td></td></tr><tr><td style='background:#CAF'>U</td><td style='background:#CAF'>n</td><td style='background:#CAF'>i</td><td style='background:#CAF'>v</td><td style='background:#CAF'>e</td></tr><tr><td style='background:#CAF'>r</td><td style='background:#CAF'>s</td><td style='background:#CAF'>i</td><td style='background:#CAF'>t</td><td style='background:#CAF'>y</td></tr></table> | |

# Problem 4 – Gosho Is Moving

Gosho studies in SoftUni. He loves it there and he is a kind of addicted to the university. So when he found out that **SoftUni is moving** to a new location, he had to do the same. So, **Gosho is moving** as well. He packed his entire luggage into **luggage pieces**. There are **3 types** of luggage pieces: **furniture**, **boxes** and **bags**. Every luggage piece goes to a different **room**, has **weight** in kilograms and a **name**. Your task is to **process the luggage pieces** according to **Gosho's filters** and present them in an **appropriate format (unordered list).**

Gosho will give you as **input** a **list of all the luggage pieces** and **4 criteria to filter** them. The **list** is a **string** that holds **many luggage pieces**, separated by **"C|\_|"** (because after moving each luggage piece he needs to drink a beer!). The **luggage piece** is a **string** in the format: **"[luggage type];[room];[name];[weight]"**, for example: "furniture;living room;pink couch;40.85kg". Below is an example for entire list of luggage pieces.

The **1st filter** may contain **1, 2, 3 or none** luggage piece **types**. The **2nd filter** is the **room**. The **3rd and 4th filters** are the **minimum and maximum weight** of all the luggage pieces of one type for one room (the **sum** **of the weights** of **all** the luggage pieces of one type **in one room** is **not less than the minimum weight and less than or equal to the max weight**).

**Note:** All the **weight's values** must be **rounded down** to an **integer** number **first**. After that all the operations with them may be executed.

### Input

The input will be read from an **HTTP GET** **request**. The **list of the luggage pieces** will be received as a string from a **text** **area** with **name '** **luggage'**.

The **1st filter** will be received from a **checkbox** with **name 'typeLuggage'**. The **2nd filter** will be received from an **input text field** with **name 'room'.** The **3rd filter** will be received from an **input text field with** name **'minWeight'.** The **4th filter** will be received from an **input text field** with **name 'maxWeight'.**

### Output

The output are three levels of nested **unordered lists (UL)** with the luggage pieces that **pass all the filters**.

The **first level** **UL** has the **types of the luggage pieces**, sorted **alphabetically** as list items. Each **list item (type)** contains a **second level** nested **UL** that has the **room names** with that type of luggage pieces, sorted **alphabetically** as list items. Each **list item (room)** contains a **third level** nested **UL** that has **only one item** – the sequence of the **names** of the luggage pieces in the room, sorted **alphabetically** and **separated by a comma and a space**. On the same line is written the **sum of the weight** of all the luggage pieces in this room. Please **follow exactly** the example below.

If any **filter is empty**, the **output** must be a **single empty** unordered list: **"<ul></ul>"**.

### Constraints

* The **luggage pieces** will be in the format "[luggage type];[room];[name];[weight]"**.** The different values are separated by **‘;’.**
* The **room** will be a string, holding the **name of the room**.
* The **weight** will always contain a **floating point** **number** and the suffix **"kg"**. The **weight** and the **suffix** are not separated by any symbols or spaces.

### Examples

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Input** | | | | | |
| luggage | furniture;living room;pink couch;40.85kgC|\_|furniture;bedroom;night table;5.12kgC|\_|boxes;kitchen;plates;10.36kgC|\_|boxes;kitchen;cups;10.36kgC|\_|boxes;kitchen;tableware;7.6kgC|\_|boxes;living room;glasses;3.32kgC|\_|boxes;living room;dresses;4.32kgC|\_|bags;hall;shoes;5.9kgC|\_| | | | | |
| minWeight | | 5 |  | typeLuggage[] | ['furniture', 'boxes', 'bags'] |
| maxWeight | | 50 |  | room | 'living room' |

|  |
| --- |
| **Output** |
| <ul><li><p>boxes</p><ul><li><p>living room</p><ul><li><p>dresses, glasses - 7kg</p></li></ul></li></ul></li><li><p>furniture</p><ul><li><p>living room</p><ul><li><p>pink couch - 40kg</p></li></ul></li></ul></li></ul> |
| **Output (formatted)** |
|  |
|
|
|