Problem 3 – Shaver Parser

Microsoft decided to invite you for a job interview (poor they :D). But they do not know the master ninja you are… or at least you claim to be one. Anyway, prove them wrong and solve the following Razor… aaaa Shaver parser.

Shaver is a modern view engine combining HTML and JavaScript in one place. In other words you have HTML-aware JavaScript. Non-technically said you can use some simple JavaScript commands to help you generate the final HTML output. This is called view and all non-HTML content (and logic) starts with **"@"** (at sign).

Here is the allowed syntax for Shaver:

- **Model properties** – you can pass an object (data model) to the view and render its properties' values in between the HTML. Each property is a key-value pair. Keys must be valid JavaScript variable names. Nested objects are not allowed. Values can be:
  - **String** – examples: "Telerik Academy", "Razor View Engine", "ASP.NET MVC"
  - **Boolean**– "true" or "false"
  - **Array** – comma separated values. Examples: "JavaScript,C Sharp,C++", "1,2,3,4"

  Syntax for rendering a value: ***@key*** – a character always follows after the invoking of the object's property. It can be " " (white space) or ">" (greater than). This character should also be rendered in the final result, even if it is white space.

  Example – model is:

  - **title**:Telerik Academy
  - **subtitle**:Free Software Trainings

View	HTML Result

`<div>`
`    <span>@title</span>`
`    <span>`
`        @subtitle`
`    </span>`
`</div>`
	`<div>`
	`    <span>Telerik Academy</span>`
	`    <span>`
	`        Free Software Trainings`
	`    </span>`
	`</div>`

- **Define section** – you can define sections with content. Each section has unique name and can contain only plain HTML elements. All sections are defined at the beginning of the Shaver view. Section defining is not part of the final result.

  Syntax is: ***@section** name content*

Example – defining a section with name "menu" and unordered list as content

```
@section menu {
<ul>
    <li>Home</li>
    <li>About us</li>
</ul>
}
```

- **Render section** – previously defined sections can be rendered multiple times on the HTML by specifying their name. Non existing sections will result in error.

  Syntax is: **@renderSection("*name*")**

  Example – rendering the "menu" section inside our view

View	HTML Result

`<div>`
`    <h1>Telerik Academy</h1>`
`    <div>`
`        @renderSection("menu")`
`    </div>`
`</div>`
	`<div>`
	`    <h1>Telerik Academy</h1>`
	`    <div>`
	`        <ul>`
	`            <li>Home</li>`
	`            <li>About us</li>`
	`        </ul>`
	`    </div>`
	`</div>`

- **Conditional statement** – conditional statement depending on Boolean property in the model object. If the condition is "true", its content is rendered, otherwise it is not. The condition cannot be Boolean expressions or directly the "true" or "false" literals. It must be a key in the model.

  Syntax is: **@if (*condition*) content**

  Example – model is

  - **masterNinja**:true
  - **notNinja**:false
  - **myDegree**:the best ninja

View	HTML Result

`<div>`
    `<h1>Telerik Academy</h1>`
    `@if (masterNinja) {`
        `<h2>I am @myDegree</h2>`
    `}`
    `<p>Very good developer</p>`
    `@if (notNinja) {`
        `<h2>I'm not @myDegree</h2>`
    `}`
`</div>`
	`<div>`
	    `<h1>Telerik Academy</h1>`
	    `<h2>I am the best ninja</h2>`
	    `<p>Very good developer</p>`
	`</div>`

- **Loop** – loops over an array value from the model object and prints the loop content for each item in the collection.

  Syntax is: **@foreach (var *item* in *collection*) content**

  Example – model is

  - **trainers**:Ivo,Niki,Dodo
  - **degree**:ninja

View	HTML Result

`<div>`
    `<h1>Telerik Academy</h1>`
    `<ul>`
        `@foreach (var trainer in trainers) {`
            `<li>@trainer is @degree</li>`
        `}`
    `</ul>`
`</div>`
	`<div>`
	    `<h1>Telerik Academy</h1>`
	    `<ul>`
	      `<li>Ivo is ninja</li>`
	      `<li>Niki is ninja</li>`
	      `<li>Dodo is ninja</li>`
	    `</ul>`
	`</div>`

- **Escaping** – when it is needed to write an actual @ (at sign) on the final result, it can be escaped with **"@@"** (two at signs).

  Example

View	HTML Result

`<div>`
    `<h1>@@Telerik Academy</h1>`
`<div>`
	`<div>`
	    `<h1>@Telerik Academy</h1>`
	`<div>`

- **Additional notes**
    - Using **eval()**, **external libraries** (or parts of them) and **regular expressions** is **not allowed**!
    - Curly brackets (**"{"** and **"}"**) will not appear anywhere outside of the commands content.
    - All opening curly brackets (**"{"**) will be on the line where the commands start.
    - All closing curly brackets (**"}"**) will have only white space (**" "**) on the line they appear.
    - There will not be any nested conditional and loop commands.
    - All commands are case sensitive.
    - All tests will contain only valid Shaver views.
    - Some tests try only some of the commands so partial solutions may give you points. Keep trying! ☺

Input

The input data should be read from the console.

On the first line you will receive **N** – the number of key-value pairs in the model object.

On the next **N** number of lines you will receive each key-value pair from the model object. The key is separated from the value with the colon (**":"**) sign. All array values will be separated with the comma **(",")** sign. There will never be colons or commas in the values themselves.

On the next line you will receive **M** - the number of lines in the view.

On the next **M** lines you will receive the view which you have to parse.

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 printed on the console.

On the first **M** lines of the output you should print the parsed HTML.

Sample solution code (in JavaScript)

```
function solve(params) {
    var s = params[0];
    var result = '';
    // Your solution here
    console.log(result);
}
```

Constraints

- **N** will be between **0** and **20**, inclusive.
- **M** will be between **1** and **200**, inclusive.
- All lines from the view will contain no more than **200** symbols.
- All views will be valid and as described.
- Allowed working time for your program: **0.10 seconds**.
- Allowed memory: **16 MB**.

Example

Input

```
6
title:Telerik Academy
showSubtitle:true
subTitle:Free training
showMarks:false
marks:3,4,5,6
students:Pesho,Gosho,Ivan
42
@section menu {
<ul id="menu">
    <li>Home</li>
    <li>About us</li>
</ul>
}
@section footer {
<footer>
    Copyright Telerik Academy 2014
</footer>
}
<!DOCTYPE html>
<html>
<head>
    <title>Telerik Academy</title>
</head>
<body>
    @renderSection("menu")

    <h1>@title</h1>
    @if (showSubtitle) {
        <h2>@subTitle</h2>
        <div>@@JustNormalTextWithDoubleKliomba ;)</div>
    }

    <ul>
        @foreach (var student in students) {
            <li>
                @student
            </li>
            <li>Multiline @title</li>
        }
    </ul>
    @if (showMarks) {
        <div>
            @marks
        </div>
    }

    @renderSection("footer")
</body>
</html>
```

Output

```
<!DOCTYPE html>
<html>
<head>
    <title>Telerik Academy</title>
</head>
<body>
    <ul id="menu">
        <li>Home</li>
        <li>About us</li>
    </ul>

    <h1>Telerik Academy</h1>
    <h2>Free training</h2>
    <div>@JustNormalTextWithDoubleKliomba ;)</div>

    <ul>
        <li>
            Pesho
        </li>
        <li>Multiline Telerik Academy</li>
        <li>
            Gosho
        </li>
        <li>Multiline Telerik Academy</li>
        <li>
            Ivan
        </li>
        <li>Multiline Telerik Academy</li>
    </ul>

    <footer>
        Copyright Telerik Academy 2014
    </footer>
</body>
</html>
```