Problem 3 – XML Messenger

XML may not see much use these days, but there is legacy software out there that needs maintenance. You are tasked by your boss to add an HTML parser to an old messaging app. The catch – it has to support all of its versions!

Write a JavaScript program that parses messages from XML format to HTML for display on a web page. A message has a body (the text) and attributes, which contain metadata – who sent the message, and who it is targeted at. This is the general format:

```
<message to="Alice" from="Bob">This is a test</message>
```

A valid message will **always be enclosed** in an opening and closing `<message>` tags and will have **no extra characters** before the opening and after the closing tag. Inside the opening tag, there will be only valid attributes. If the message does not match this format or not all characters belong to an attribute, print "`Invalid message format`".

The opening tag contains the **attributes**. An attribute is in the form `key="value"`, where the **key** will only contain **lowercase Latin letters** and the **value** will contain **Latin alphanumeric characters, spaces and dots**. Both key and value must be **at least one character long**. Characters which do not belong to a valid attribute must not be accepted – such messages are **invalid**. If the message format is valid, but it doesn't have both valid `to` and `from` attributes, print "`Missing attributes`".

Messages come from different versions of the software, so they **may have additional attributes**, but we only care about the sender and recipient. The `to` and `from` attributes **may appear in different order** along with other attributes, which we ignore – the message is still **valid**.

The body of the message **may contain any character**, including line breaks (`\n` control symbol) – there is no need to validate it. Every time you encounter a line break, you must print the following text in a new paragraph. **See the examples for details**.

The following message are **all valid**:

```
<message to="Bob" from="Alice" timestamp="1497254092">Hey man, what's up?</message>
<message from="Ivan Ivanov" to="Grace">Not much, just chillin. How about
you?</message>
<message to="MasterBlaster" from="TheAnimal" timestamp="1497254114">Same old, same
old\nLet's go out for a beer</message>
<message version="2.0" to="Alice" from="Charlie">TGIF!</message>
```

The sender of the message is highlighted in green, while the recipient is in blue. Note the order may be mixed and the additional attributes are just ignored (but are still in valid key-value format).

The following messages are **all invalid** and an **error message** must be printed, as instructed above:

Message	Reason
`<message from="Alice" timestamp="1497254112">This is a test</message>`	missing `to` attribute
`<message to="Matilda" from="Charlie"><media type="image" src="slyfox.jpg"/></message><meta version="2.0"/>`	extra characters after closing tag
`<message from="MasterBlaster" to="TheAnimal" color="#FF340B">FWD: Funny stuff</message>`	invalid characters in attribute (don't form a valid attribute)
`<message from="Hillary" to="Edward" secret:true>VGhpcyBpcyBhIHRlc3Q</message>`	invalid characters in opening tag (don't form a valid attribute)

After you parse the message and extract the sender, recipient and message body, print the HTML-formatted result:

```
<article>
  <div>From: <span class="sender">Alice</span></div>
  <div>To: <span class="recipient">Bob</span></div>
  <div>
    <p>Same old, same old</p>
  </div>
</article>
```

Note that if there is a **line break** in the message body, you have to add an additional **paragraph**. Nested elements are **indented** by two spaces for each level. **See the examples for more details**.

Input

You will receive a **single string** as input – a message that needs to be **validated**.

Output

Print on the **console** the HTML-formatted **result** or an **error message**, depending on the validation.

Constraints

- The message body will not contain XML brackets (**<>**), there is no need to validate it
- There may be any number of spaces before and after an attribute
- An attribute will **not** contain spaces before or after the equal sign

Examples

Input
`<message to="Bob" from="Alice" timestamp="1497254092">Hey man, what's up?</message>`
Output

```
<article>
  <div>From: <span class="sender">Alice</span></div>
  <div>To: <span class="recipient">Bob</span></div>
  <div>
    <p>Hey man, what's up?</p>
  </div>
</article>
```

Input
`<message from="John Doe" to="Alice">Not much, just chillin. How about you?</message>`
Output

```
<article>
  <div>From: <span class="sender">John Doe</span></div>
  <div>To: <span class="recipient">Alice</span></div>
  <div>
    <p>Not much, just chillin. How about you?</p>
  </div>
</article>
```

Input

```
<message from="Alice" timestamp="1497254112">Same old, same old</message>
```

Output

```
Missing attributes
```

Input

```
<message to="Bob" from="Alice" timestamp="1497254114">Same old, same old
Let's go out for a beer</message>
```

Output

```
<article>
  <div>From: <span class="sender">Alice</span></div>
  <div>To: <span class="recipient">Bob</span></div>
  <div>
    <p>Same old, same old</p>
    <p>Let's go out for a beer</p>
  </div>
</article>
```

Note

Note the line break in the input message – the body has an extra paragraph! To test locally, place a **\n** control symbol in the string.

Input

```
<message to="Alice" from="Charlie"><img src="fox.jpg"/></message><meta version="2"/>
```

Output

```
Invalid message format
```

Input

```
<message from="Hillary" to="Edward" secret:true>VGhpcyBpcyBhIHRlc3Q</message>
```

Output

```
Invalid message format
```