

# Attribute Parser

We have defined our own markup language *HRML*. In HRML, each element consists of a starting and ending tag, and there are attributes associated with each tag. Only starting tags can have attributes. We can call an attribute by referencing the tag, followed by a tilde, '~' and the name of the attribute. The tags may also be nested.

The *opening tags* follow the format:

`<tag-name attribute1-name = "value1" attribute2-name = "value2" ... >`

The *closing tags* follow the format:

`< /tag-name >`

For example:

```
<tag1 value = "HelloWorld">
<tag2 name = "Name1">
</tag2>
</tag1>
```

The attributes are referenced as:

```
tag1~value
tag1.tag2~name
```

You are given the source code in HRML format consisting of  $N$  lines. You have to answer  $Q$  queries. Each query asks you to print the value of the attribute specified. Print *"Not Found!"* if there isn't any such attribute.

## Input Format

The first line consists of two space separated integers,  $N$  and  $Q$ .  $N$  specifies the number of lines in the HRML source program.  $Q$  specifies the number of queries.

The following  $N$  lines consist of either an opening tag with zero or more attributes or a closing tag.

$Q$  queries follow. Each query consists of string that references an attribute in the source program.

## Constraints

$$1 \leq N \leq 20$$

$$1 \leq Q \leq 20$$

Each line in the source program contains, at max, **200** characters.  
Every reference to the attributes in the  $Q$  queries contains at max **200** characters.  
All tag names are unique.

## Output Format

Print the value of the attribute for each query. Print *"Not Found!"* without quotes if there is no such attribute in the source program.

## Sample Input

```
4 3
<tag1 value = "HelloWorld">
<tag2 name = "Name1">
</tag2>
</tag1>
tag1.tag2~name
tag1~name
tag1~value
```

## Sample Output

```
Name1
Not Found!
HelloWorld
```