Brenden Politeness Filter

Brenden is a good guy, he wants to make a filter to be put in class rooms, so then no one whould be able to throw bad words to others.

He invented a politeness filter, which could be attached to the ceil of the class room, and it will allow only good words to pass.

How the filter works?

The politeness filter contains several steps of filtering(layers) , each step(layer) would test the word against its own filtering settings, then send the word into its queue if the word matched its filtering string, the queue will forward those words to sepcific group of people.

In another way, the configuration consist of several consequtive QUEUE tags, each one has a name,bpf properties:

```
<QUEUE  name="queue_name"
        bpf = "bpf_string"
/>
```

As you know, talking is not like shouting, so the filter will distinguish between normal words, and shouted words.

The bpf string is written in the following grammer:

```
<bpf_string>         ::= "" | <words_filter> "|| (shout and " <words_filter> ")"
<words_filter>       ::= "(" <many_words_filter> ")"
<many_words_filter>  ::= <single_word_filter> | <single_word_filter> " or " <single_word_filter>
<single_word_filter> ::= "word " <word>
```

(Please review the Backus–Naur form https://en.wikipedia.org/wiki/Backus%E2%80%93Naur_form if you didn't understand the notation).

(Note: if the notation is a little bit hard, don't worry, the example is clear)

Ok, Your task now is to write the configuration of a politeness filter!

The input will start with the word "Config:" then following 4 lines with the following grammer:

```
<line>                ::= "      - " <queue_name> "=" <words_specifications>
<words_specifications> ::= "" | <words_list>
<words_list>          ::= <word> | <word> "," <words_list>
```

Please note that <queue_name> and <word> are strings that consists of combinations of lower case, upper case, numbers, underscores only (no other punctuations, no spaces), with a limit of 50 character each, and the <word_list> is no more than 10 words.

Also, you can assume that queue names are different, also the <word_specificatinos>s are different.

Final note, if the <words_specifications> was empty, don't print the queue at all.

And after printing all the queues, please print a final queue with queue_name="OTHER" and an empty bpf.

Ok, that's all, here is some examples:

Example1

Input

Config:

- BPF_WORDS_GOOD=Palastine,Syria
- BPF_WORDS_SECRET=
- BPF_WORDS_BAD=Materialism,Racism
- BPF_WORDS_POLITICAL=tyranny

Output

```
<QUEUE  name="BPF_WORDS_GOOD"
            bpf = "(word Palastine or word Syria) || (shout and (word Palastine or word Syria))"
/>
<QUEUE  name="BPF_WORDS_BAD"
            bpf = "(word Materialism or word Racism) || (shout and (word Materialism or word Racism))"
/>
<QUEUE  name="BPF_WORDS_POLITICAL"
            bpf = "(word tyranny) || (shout and (word tyranny))"
/>
<QUEUE  name="OTHER"
            bpf = ""
/>
```

Example2

Input

Config:

- BPF_WORDS_1=1111,2222
- BPF_WORDS_2=3333,4444,4455,4466
- BPF_WORDS_3=5555
- BPF_WORDS_4=7777,8888

Output

```
<QUEUE  name="BPF_WORDS_1"
            bpf = "(word 1111 or word 2222) || (shout and (word 1111 or word 2222))"
/>
<QUEUE  name="BPF_WORDS_2"
            bpf = "(word 3333 or word 4444 or word 4455 or word 4466) || (shout and (word 3333 or word 4444 or word
4455 or word 4466))"
/>
<QUEUE  name="BPF_WORDS_3"
            bpf = "(word 5555) || (shout and (word 5555))"
/>
<QUEUE  name="BPF_WORDS_4"
            bpf = "(word 7777 or word 8888) || (shout and (word 7777 or word 8888))"
/>
<QUEUE  name="OTHER"
            bpf = ""
/>
```