A. Complete DFSA to RegExp Translator

time limit per test: 1 second
memory limit per test: 16 megabytes
input: input.txt

Implement a DFSA to RegExp Translator. Given an DFSA description in the input.txt (see input file format) your program should output to console an error description (see validation errors) OR a regular expression that corresponds to the given DFSA. The regular expression should be built according to a slightly modified version of the Kleene's algorithm (see Kleene's algorithm).

Input

states=[s1,s2,...] // s1 , s2, ... ∈ latin letters, words and numbers

alpha=[a1,a2,...] // a1 , a2, ... ∈ latin letters, words, numbers and character '_'(underscore)

initial=[s] // s ∈ states

accepting=[s1,s2,...] // s1, s2 ∈ states

trans=[s1>a>s2,...] // s1,s2,...∈ states; a ∈ alpha

Examples

input Copy

```
states=[on,off]
alpha=[turn_on,turn_off]
initial=[off]
accepting=[on]
trans=[off>turn_on>off,on>turn_off>on]
```

output Copy

```
E6: Some states are disjoint
```

input Copy

```
states=[0,1]
alpha=[a,b]
initial=[0]
accepting=[1]
trans=[0>a>0,0>b>1,1>a>1,1>b>1]
```

output Copy

```
(((a|eps)(a|eps)*(b)|(b))(({})(a|eps)*(b)|(a|b|eps))*(({})(a|eps)*(b)|
(a|b|eps))|((a|eps)(a|eps)*(b)|(b)))
```

input Copy

```
states=[on,off]
alpha=[turn_on,turn_off]
initial=[off]
accepting=[]
trans=[off>turn_on>on,on>turn_off>off]
```

output Copy

```
E3: Set of accepting states is empty
```

Note

Errors: The errors may appear in the inputs, which should lead to error message according to the priority given below. Only 1 error message should be shown, if required. It should be assumed that for each line read from the inputs all the possible errors should be checked in the given priority, if applicable for the current line.

- E1: Input file is malformed
- E2: Initial state is not defined
- E3: Set of accepting states is empty
- E4: A state 's' is not in the set of states
- E5: A transition 'a' is not represented in the alphabet

- E6: Some states are disjoint
- E7: FSA is nondeterministic

Kleene's Algorithm:

The Kleene's Algorithm should be used as presented in the Lab 10, but with following modifications:

- Denote ∅ as {}
- Denote Ɛ as eps
- Define update rule with the additional parentheses:
  $R_{ij}^k = (R_{ik}^{k-1})(R_{kk}^{k-1}) * (R_{kj}^{k-1})|(R_{ij}^{k-1})$
- At each step each regular expression should be surrounded by parentheses, e.g.
  $R_{ij}^k = ((a|eps)(a|eps) * (a|eps)|(a|eps))$
- Optimization of the regular expressions should NOT be done
- The regular expression parts' content should be in lexicographical order, but Ɛ should be at the end of each part, e.g. $(a|b|eps)*$
- Assume that input files and standard outputs should end with a new line character

204971121	May/07/2023 15:22	Partial result: 9.75 points
204970930	May/07/2023 15:20	Partial result: 9.75 points
204970344	May/07/2023 15:15	Partial result: 9.75 points

→ Your points

	Points
A	15
Total	15