Look at the following FST, implemented using Pynini:

```
import pynini as pn
fst = (pn.a("a") | pn.a("e")) + pn.t("a", pn.a("0").closure(0,5)) | pn.t(pn.a("a").star, "0") + pn.a("xxx")
```

You may use a computer to answer the following questions:

1. Give a regular expression that is equivalent to the input language of this finite state transducer. (1 point)
2. Give inputs that are mapped by this FST to 0 outputs, 1 output, 2 outputs, and more than 2 outputs. (4 points)
3. Make program that uses Pynini to map numbers (represented as strings) to French words (up to 999999999). Use provided numbers2words.py as starter.

It's a good idea to implement this with two FSTs. The first FST will factorize a number into into sequences annotated with powers of ten:

```
0    -> 0
1    -> 1
10   -> 1^
23   -> 2^ 3
203 ->  2^^ 3
```

A second FST will convert the factorized form into words:

```
0    -> zero
1    -> un
1^   -> dix
2^ 3   -> vingt et un
```

Finally, you should also handle decimals. Decimals should be verbalized digit-by-digit: 0.046 whould be converted into "zero virgule zero quatre six".

You are not expected to know French to complete this task. Use materials on the web, e.g. https://tuto.pages-informatique.com/writing-out-numbers-in-french-letters.php

Yes, this does give some advantage to the French students. However, the situation is quite realistic: when you work in NLP, you often have to deal with a language that you don't know.

You should submit the program, which is a modification of the provided template. It should run as is, without any additional dependencies. Scoring: an undisclosed test set (similar to the provided unit test) will be used for scoring. Depending on the number of successful and failed tests, you will get 0 to 10 points. The scale is linear: `score = 10.0 * num_successful_tests / num_tests`