Write a bare-bones Markov text generator.

Implement a function of the form
`finish_sentence(sentence, n, corpus, deterministic=False)`
that takes four arguments:

1. a `sentence` [list of tokens] that we're trying to build on,
2. `n` [int], the length of n-grams to use for prediction, and
3. a source `corpus` [list of tokens]
4. a flag indicating whether the process should be `deterministic` [bool]

and returns an extended sentence until the first `.`, `?`, or `!` is found OR until it has 15 total tokens.

If the input flag `deterministic` is true, choose at each step the single most probable next token. When two tokens are equally probable, choose the lesser one (according to Python).

If `deterministic` is false, draw the next word randomly from the appropriate distribution. Use stupid backoff and no smoothing.

Demonstrate the use of your function in both deterministic and stochastic modes.

As one (simple) test case, use the following inputs:

```
sentence = ['she', 'was', 'not']
n = 3
corpus = [
    w.lower for w in
    nltk.corpus.gutenberg.words('austen-sense.txt')
]
deterministic = True
```

The result should be
`['she', 'was', 'not', 'in', 'the', 'world', '.']`