What is the most "likely" word of length n?

Construct the letter transition matrix implied by a text corpus, including the invisible *word-break* letter. The function `transition_matrix()` should take a corpus (list of words `\[a-z]+\`) and return a 27×27 numpy array, starting with 'a'.
Important: add 1 to each transition count before normalization so that you never have $p(q_{t+1}|q_t) = 0$.

Use the transition matrix to compute the likelihood of each word (the product of its $n+1$ transition probabilities). The function `most_likely_word()` should take

1. a dictionary (list of words)
2. the transition matrix
3. n

and return the word(s) of length n with the highest likelihood.
Hint: Do the computation in log scale to avoid computational issues.

Put both functions in a file titled `hw07_solution.py`.

Run `hw07_evaluate.py`, making sure that your solution file is on the Python path. If you're unsure whether the result is satisfactory, ask the instructor or TA.