QUESTIONS After you've written the program, use it to examine the output you get with bi-grams and tri-grams and answer the following questions:

Question 1 (10pts) For both bi-grams and tri-grams, give 5 examples of words that look especially unnatural. For each word, explain why the model generates these unnatural strings.

Bigrams:

1. # L D #
    a. P(L|#) is high, so the string starts with L. P(D|L) is high as well, and as stated before, D seems to be a common ending phone since P(#|D) is high. This string could be found at the end of a word, but it is missing a vowel at the beginning because the model does not account for context further than the preceding letter.
2. # D #
    a. P(D|#) is high, meaning that D is a common starting letter. But P(#|D) is also high, meaning D is also a common ending phone. This results in the bigram model producing a string that both starts and ends with D.
3. # N K #
    a. P(K|#) is high, so the string starts with K. P(N|K) is high as well, and K seems to be a common ending phone since P(#|K) is high. Once again, this is a string that could be found at the coda of a word, but it seems to be missing a vowel preceding it because the model does not account for context further than the preceding phone.
4. # B #
    a. As with the second example, P(B|#) and P(#|B) are both high, meaning that B is a common starting and ending phone. This results in the bigram model producing a string that both starts and ends with B.
5. # L Z #
    a. P(L|#) is high, so the string starts with K. P(Z|L) is high as well, and the string ends with Z since P(#|Z) is high. This is because the bigram does not account for context earlier than the last seen phone; so even though this might be a valid consonant cluster in the coda of a word, it lacks a vowel that might make it sound more English-like.

Overall, the model seems to often produce strings that are just one or two consonants with no vowels present, making them seem unnatural.

Tri-grams:

1. # # F EH S T AH L IH NG B AO L T ER AY Z AH M AH N K L AE SH AH N IH Z AH L #
    a. This word seems excessively long to me to sound natural. I feel that if it were broken down into more parts, the separate pieces would sound English-like. The

model could be generating words like this because it continues to add phones as long as valid trigrams exist and it does not have any concept of word length.

2. # # D AH M EH N S AH K W AH DH ER W OW L V IY #
    a. Again, this word seems weirdly long to me, like a mash-up of actual English words. There are no invalid combinations of consonants but the model seems to still be appending phones to strings as long as the # symbol is not being generated.
3. # # T AE T AH N Z AY L AH NG K #
    a. I found the string ending in “NG K” to be strange and unnatural as I cannot think of any English words that have a [k] following an ngma. This could be because despite the longer history, the model still does not have awareness of phonotactic constraints and does not know which clusters are acceptable in English.
4. # # V AY B D #
    a. Again, I found the ending “B D” to be not very English-like as I do not think that coda appears anywhere in English. I feel that the rest of the string is fine except for the last phone at the end. The cluster may have been seen in a strange rare case in the training, and the model produced this string as it is not trained to handle illegal cases of consonant clusters.
5. # # K AE SH T #
    a. As with the previous two examples, I found that “SH T” was a weird cluster that is not very English-sounding. These cases of weird clusters were rare, but I believe that they are caused by irregular cases that appear in training, resulting in the model producing these words as it lacks awareness of the phonotactic constraints of English.

Question 2 (3pts) Which model (the bi-gram or tri-gram) generates words that look more natural and English-like? Why is this model better able to produce English-like words? Be specific!

The trigram model generates words that look more natural and English-like, as it does not produce words that are blatantly unnatural, such as words with only one or two consonants like “# N K #” and “# B #”. The bigram model has this issue as it only works with the last seen phone. It appears to be able to generate valid consonant clusters, but not so much for well-formed, natural-sounding English words. Because the trigram model works with a longer history, it performs better with generating English-like words and avoids generating words that seem to be missing parts, such as a vowel. As a result, the trigram model produces unnatural-sounding words noticeably less than the bigram model.

Question 3 (6) Use smoothed bigrams and trigrams to find the perplexity of X.txt and Y.txt. What is the perplexity of each corpus according to each model? How should the results be interpreted (that is, what does each model say about how English-like the two corpora are)?

	X.txt	Y.txt

Bigrams	13.933330128966505	62.88960430952193
Trigrams	14.841225994530822	41.1752759854547

Based on the results, it can be assumed that both models say that X.txt is more English-like than Y.txt, as their perplexities are always lower for the former than the latter text file.

Question 4 (6) Look at the probabilities assigned to each word in the X and Y files by the two models. A model does a good job distinguishing between English-like words from non-English-like words if it is possible to choose a threshold such that the probabilities of all words in one file are below this threshold and the probabilities of all words are above it in the other file. Is it possible to do this for the bi-gram model? How about the tri-gram model? Which model does a better job distinguishing the two sets of words? Explain.

It is not possible to do this for the bigram model as there is overlap between the minimum and maximum probabilities for the files (minimum probability for Y.txt is 4.0725×10^(-8) and maximum probability for X.txt is 8.6512×10^(−9)).

It is still not possible to do this for the trigram model as the minimum probability for Y.txt is 1.0484×10^(-8) and maximum probability for X.txt is 3.3131×10^(−9)).

However, the trigram model still does a better job distinguishing between the two sets of words because there is less overlap between the minimum and maximum probabilities for the test files for the trigram model than for the bigram model. The trigram model struggles less with distinguishing the two sets because it accounts for more context that the bigram model.