Part I:

Given this corpus:
<s>I am Sam</s>
<s>Sam I am</s>
<s>I am Sam</s>
<s>I do not like green eggs and Sam</s>
Using a bigram language model with add-one smoothing, what is P(Sam|am)? Include <s> and </s> in your counts just like any other token.

First we transform the corpus into an all-lower-case version of itself in order to deal with the cases where words with different capitalizations may be treated as separate words in the vocabulary:

<s>i am sam</s>
<s>sam i am</s>
<s>i am sam</s>
<s>i do not like green eggs and sam</s>

From here, since we are using a bigram model, we take count of all the occurrences of each type of bigram. To do this we first find the vocabulary and count the occurrences of each word of the vocabulary in the corpus. Next, we find all length two permutations of the vocabulary, also known as the bigrams of the vocabulary, and then count the occurrences of each bigram in the corpus. Since the size of such a list is $|V|^2 = 11^2 = 121$ I will forgo presenting it here. Normally the computation would be:

$$P(Sam\ |am) = P(sam\ |am) = \frac{Count(am, sam)}{Count(am)}$$

But since we are using add-one smoothing we will be using the following:

$$P(Sam\ |\ am) = P(sam\ |\ am) = \frac{Count(am, sam) + 1}{Count(am) + |V|}$$

The idea behind add-one smoothing is to distribute probability mass to occurrences we do not see in the corpus. We do this because they may still be valid bigrams even if we have not seen them. To accomplish this, we add one to the count of each bigram, and when we do the computation, we add the size of the vocabulary to the denominator. This works because we are adding |V| occurrences for each word context when adding one to each count. Using all of this we can see that we have:

$$Vocabulary\ V = \{i, am, sam, do, not, like, green, eggs, and, < s >, </s >\}, \qquad and \qquad |V| = 11$$

$$\frac{Count(am, sam) + 1}{Count(am) +\ |V|} = \frac{2+1}{3+11} = \frac{3}{14}$$

Part II

1. There are 39502 word types, exclusive of '<s>', in the training corpus after the pre-processing steps.
2. There are 2221290 tokens, exclusive of '<s>' tokens, in the training corpus after the pre-processing steps.
3. By method of counting my program observed 1248 word types in the test data, excluding '<s>', of which 49 were not in the training corpus. This means there were roughly 3.926% word types in the test data that were not in the training data. My program also observed a 2769 tokens in the test data, excluding '<s>', of which 50 were not in the training corpus. This means there were roughly 1.086% tokens in the test data that were not in the training data.
4. After replacing the singletons in the training corpus with '<unk>' and mapping the test tokens not seen in the training data to '<unk>' my program observed 2359 bigram types in the test data, with 616 bigram types not appearing in the training corpus. This means that roughly 26.113% of bigram types in the test data were not seen in the training corpus. My program also observed 2769 bigram tokens in the test data, with 618 not appearing in the training corpus. This means that roughly 22.319% of bigram tokens in the test data were not seen in the training corpus.
5. I used the following formulas to compute the log weight probabilities of the given sentence:

$$UnigramMLE(S) = \sum_{s_i \in S} log_2(\frac{count(s_i)}{|T|}), where\ |T|\ is\ the\ size\ of\ the\ corpus, and\ s_i\ is\ a\ unigram$$

$$BigramMLE(S) = \sum_{s_i s_{i+1} \in S} log_2\left(\frac{count(s_i, s_{i+1})}{count(s_i)}\right), unigrams\ s_i, and\ bigrams\ s_i s_{i+1}$$

$$BigramAddOne(S) = \sum_{s_i s_{i+1} \in S} log_2\left(\frac{count(s_i, s_{i+1}) + 1}{count(s_i) + |V|}\right), for\ unigrams\ s_i, bigrams\ (s_i, s_{i+1}), and\ |V| for\ vocab\ V$$

Where each count is what was observed in the training corpus, and vocab includes all tokens except '<s>'.

This yielded the following results:

Unigram MLE Log Probability of Given Sentence:

i :	-8.400191146154114
look :	-11.982303986215943
forward :	-12.375607193140258
to :	-5.540690135458277
hearing :	-13.505537497185392
your :	-10.953683308276174
reply :	-17.623534706583843
. :	-4.811868102497583
</s> :	-4.62532894422938
Total:	-89.81874501974097

Bigram MLE Log Probability of Given Sentence:

<s> i :	-5.631088893700847	
i look :	-8.875420257009424	
look forward :	-4.345774836841731	
forward to :	-2.2644156362321546	
to hearing :	-13.2203480948755	
hearing your :	-inf	//this had probability of 0
your reply :	-inf	//this had probability of 0
reply . :	-inf	//this had probability of 0
. </s> :	-0.0848872391724313	
Total:	-inf	

Bigram Add-One Smoothed Log Probability of Givn Sentence:

<s> i :	-6.1552832892341485
i look :	-11.584899882389982
look forward :	-10.482231743023195
forward to :	-8.825428620790015
to hearing :	-13.82744195061677
hearing your :	-15.276633330886703
your reply :	-15.31000916584291
reply . :	-15.270076277195477
. </s> :	-0.6693560071854955
Total:	-97.4013602671647

6. To compute the perplexity of the given sentence under each of the models we use the following equation:

$$Perplexity(s) = 2^{-\frac{1}{M} logWeightProbability(s)}, for\ token\ count\ of\ s = M$$

This yielded the following results:

Unigram MLE Perplexity of Given Sentence:	1009.8046830192555
Bigram MLE Perplexity of Given Sentence:	inf
Bigram Add-One Smoothed Perplexity of Given Sentence:	855.2106605432591

7. To compute the perplexity of the entire test corpus we use the following equation for each of the language models:

$$Perplexity(T) = 2^{-\frac{1}{M}\sum_{s\in T} logWeightProbability(s)}, for\ token\ count\ of\ T = M$$

This yields the following results:

Unigram MLE Perplexity of Test Data:	1082.014772901339
Bigram MLE Perplexity of Test Data:	inf
Bigram Add-One Smoothed Perplexity of Test Data:	1807.233786592602

Since we map all tokens that do not appear in the training data to '<unk>' and map all singletons in training corpus to '<unk>' this artificially decreases the perplexity of the unigram MLE results. It assigns a much larger probability mass than what may be appropriate for the outliers/singletons in the training corpus. It also means that unigrams that never appeared in the training corpus are given a significant probability. The Bigram MLE model is not afforded this luxury. Since the number of possible bigrams is so much larger, even when we map all tokens not seen in the training data to '<unk>' we may still encounter a bigram with 0 probability. Indeed, in the answer to question (4) we see that roughly 22% of bigrams in the test data did not appear in the training data, which is a significant amount and will result in a perplexity of infinity. For the Bigram Add-One smoothed model, we see a higher perplexity than the Unigram MLE model. This may be because the Bigram Add-One smoothed model does not benefit as much from mapping the unseen data in test to '<unk>' as the Unigram MLE model does. With the Unigram MLE model all '<unk>' get the same inflated aggregate value, but things are more dynamic for bigrams under the Bigram Add-One smoothed model. There are many more bigrams under the Bigram Add-One smoothed model than the Unigram MLE model, so the probability mass for '<unk>' will be spread thinner. This results in lower probabilities for the Bigram Add-One smoothed model, and thus a higher perplexity. We also note that a significantly larger probability of bigrams are missing from the test data than unigrams, namely roughly 22% versus 1.1% respectively. This indicates that the Bigram Add-One Perplexity is affected much more by low probability bigrams than the Unigram MLE model is. In conclusion, due to the overrepresentation of '<unk>' in training corpus, and the prevalence of low probability bigram in the test corpus we see a bigram perplexity larger than the unigram perplexity.