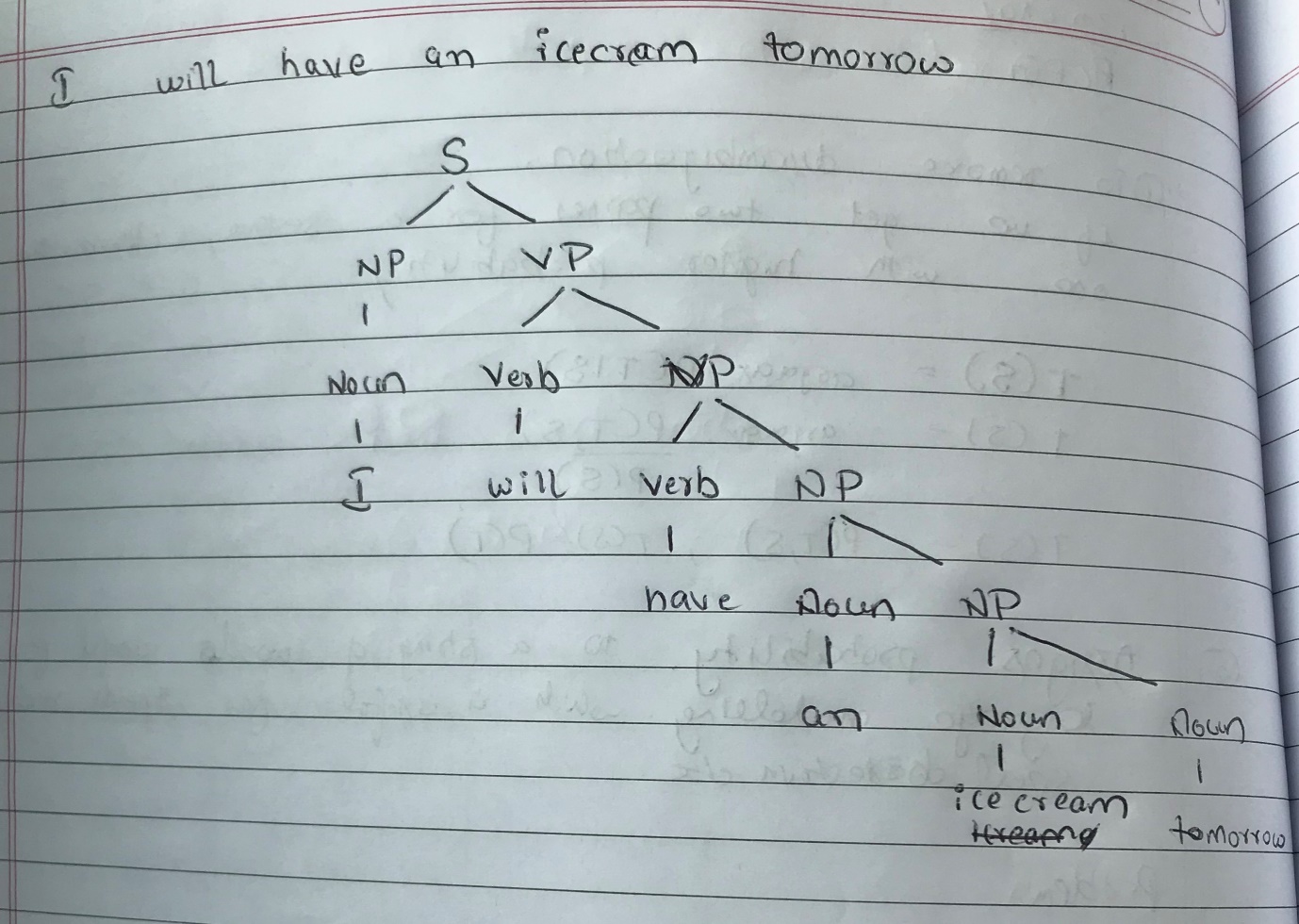
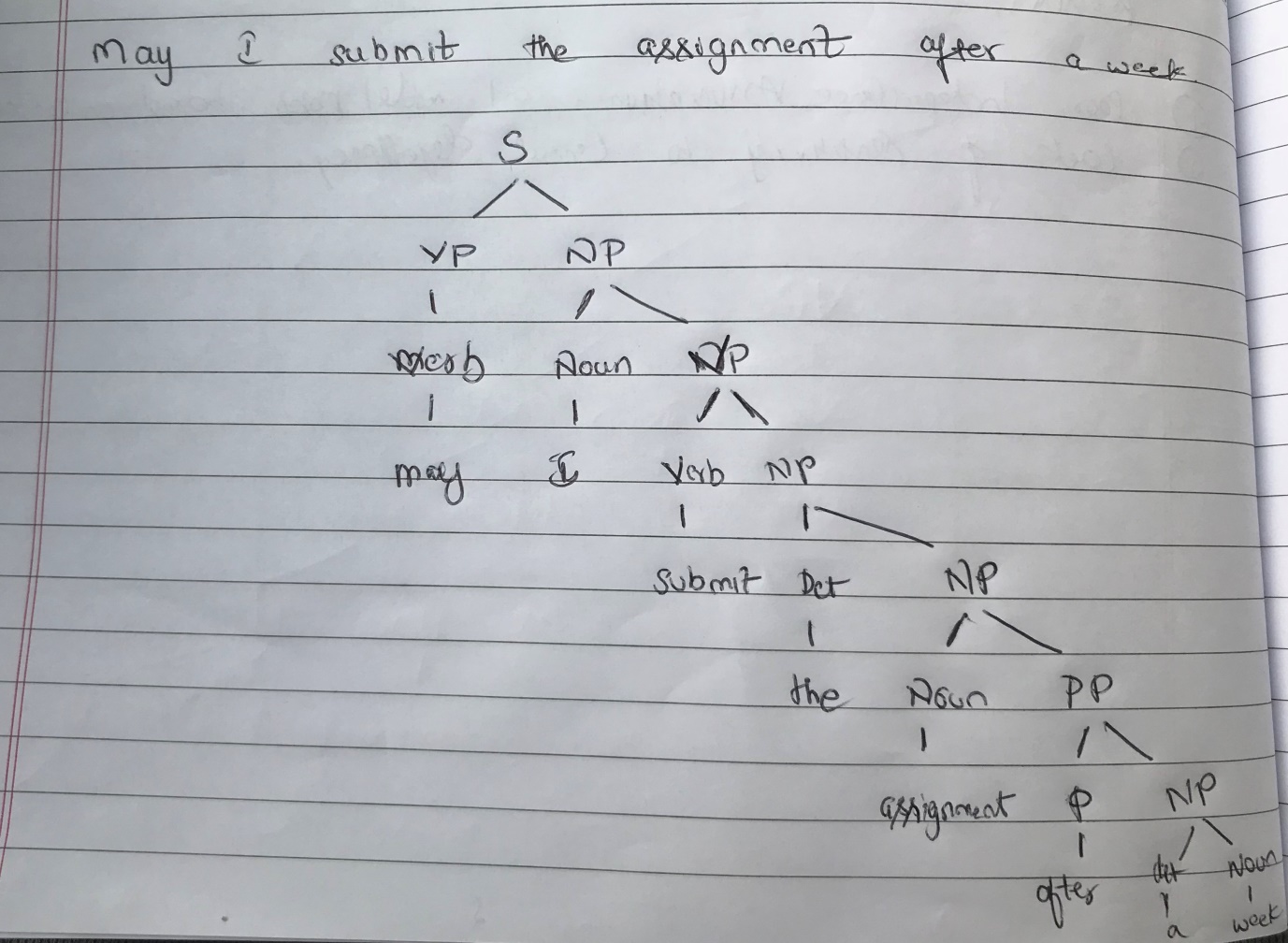
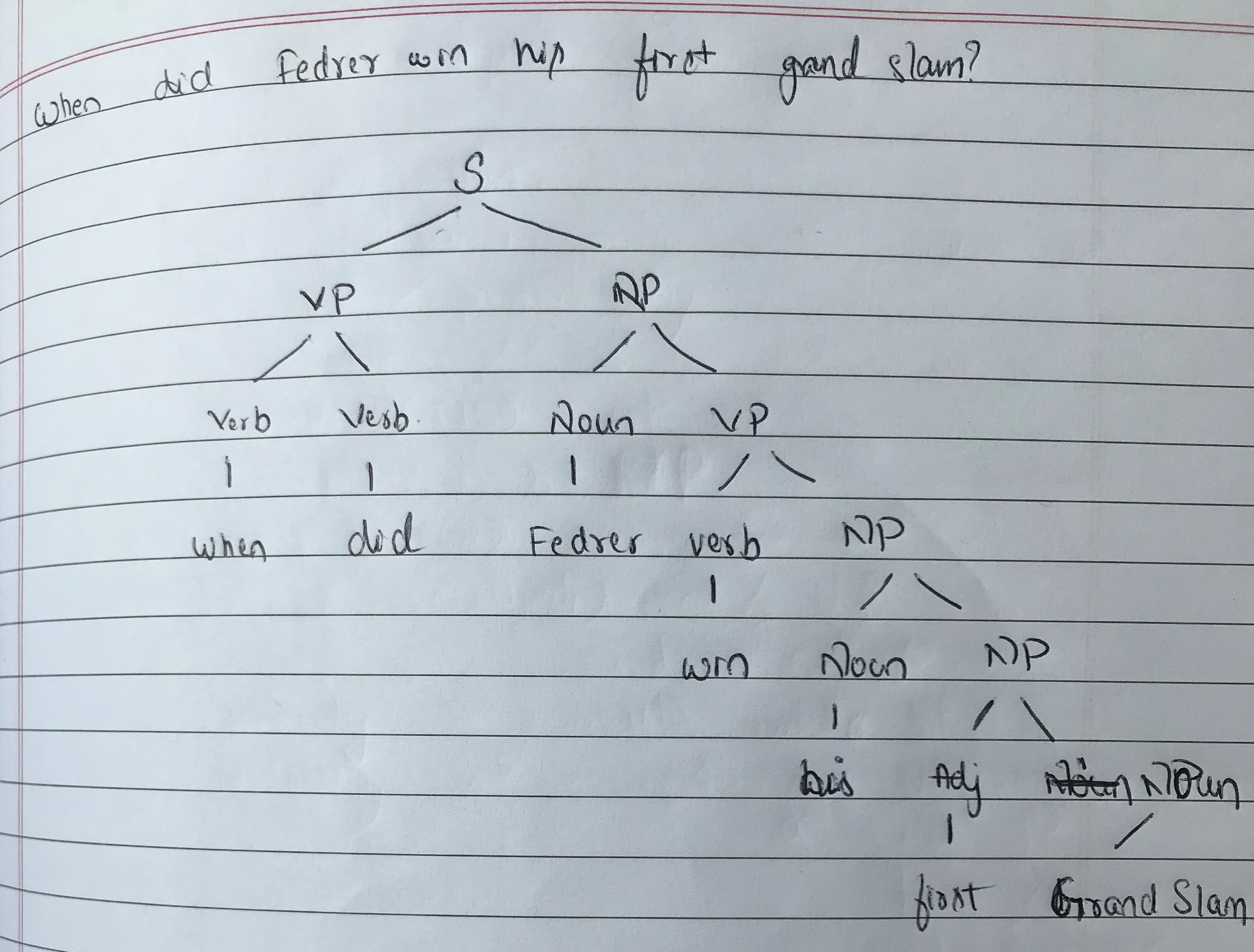
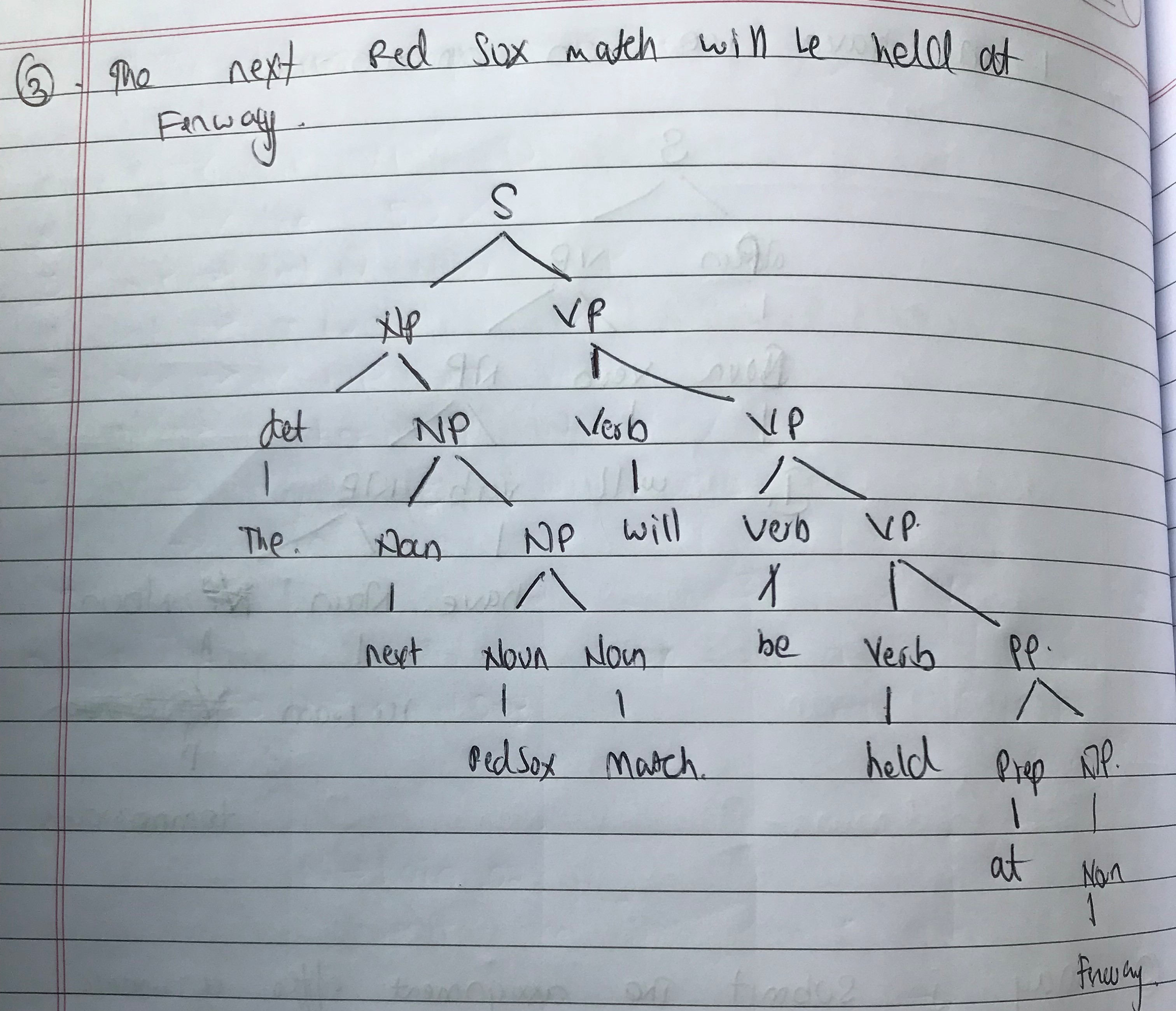
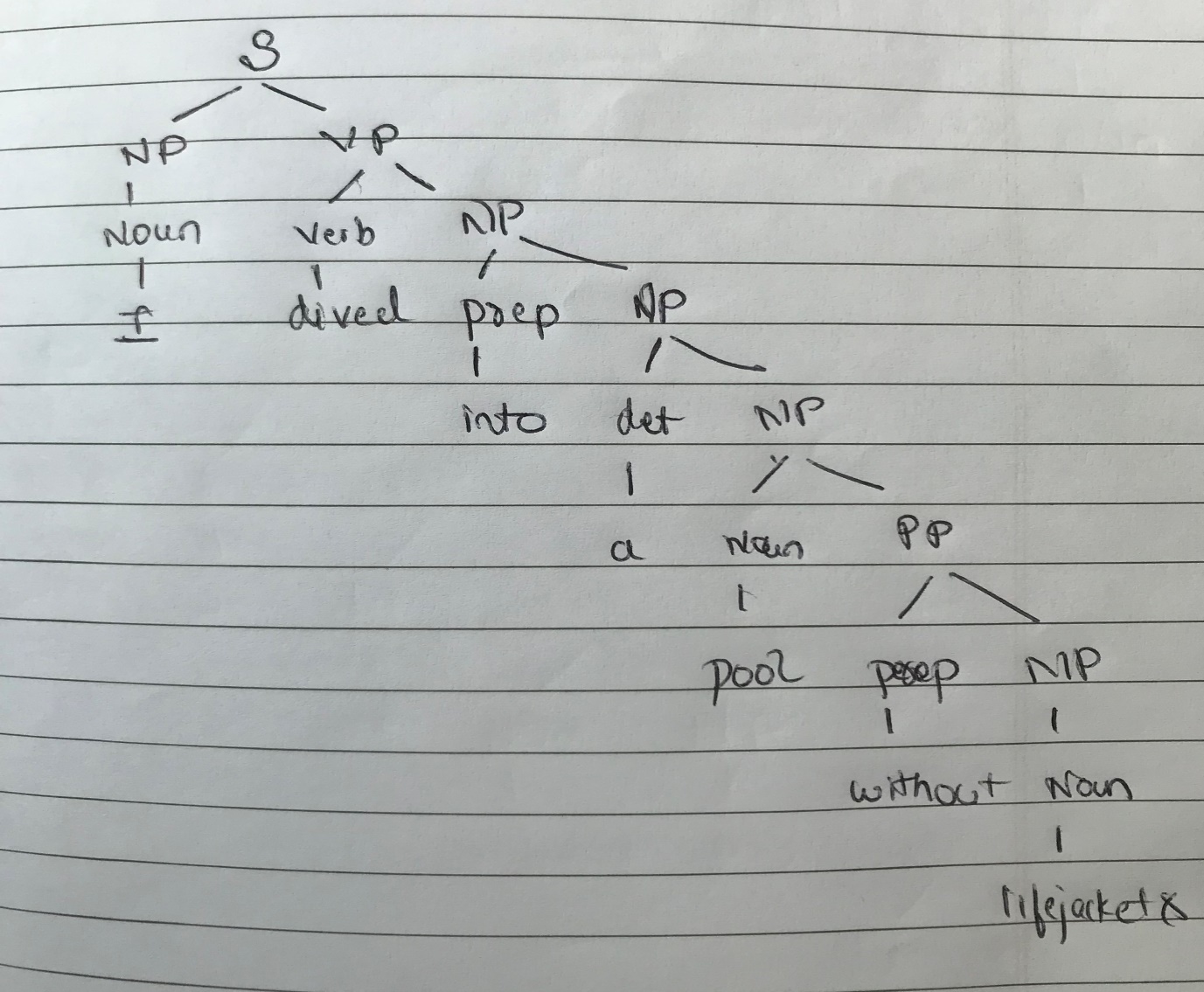
Q1. 







Question 2:

We can use a probabilistic parser to deal with input that may be incorrect.

Probabilistic context free grammar(PCFG) is used which assigns a probability to a string of words constituting a grammar. We can also assign probability to substrings of the sentence.

We can use Penn treebank, a parsed corpus that annotates syntactic or semantic sentence structure.

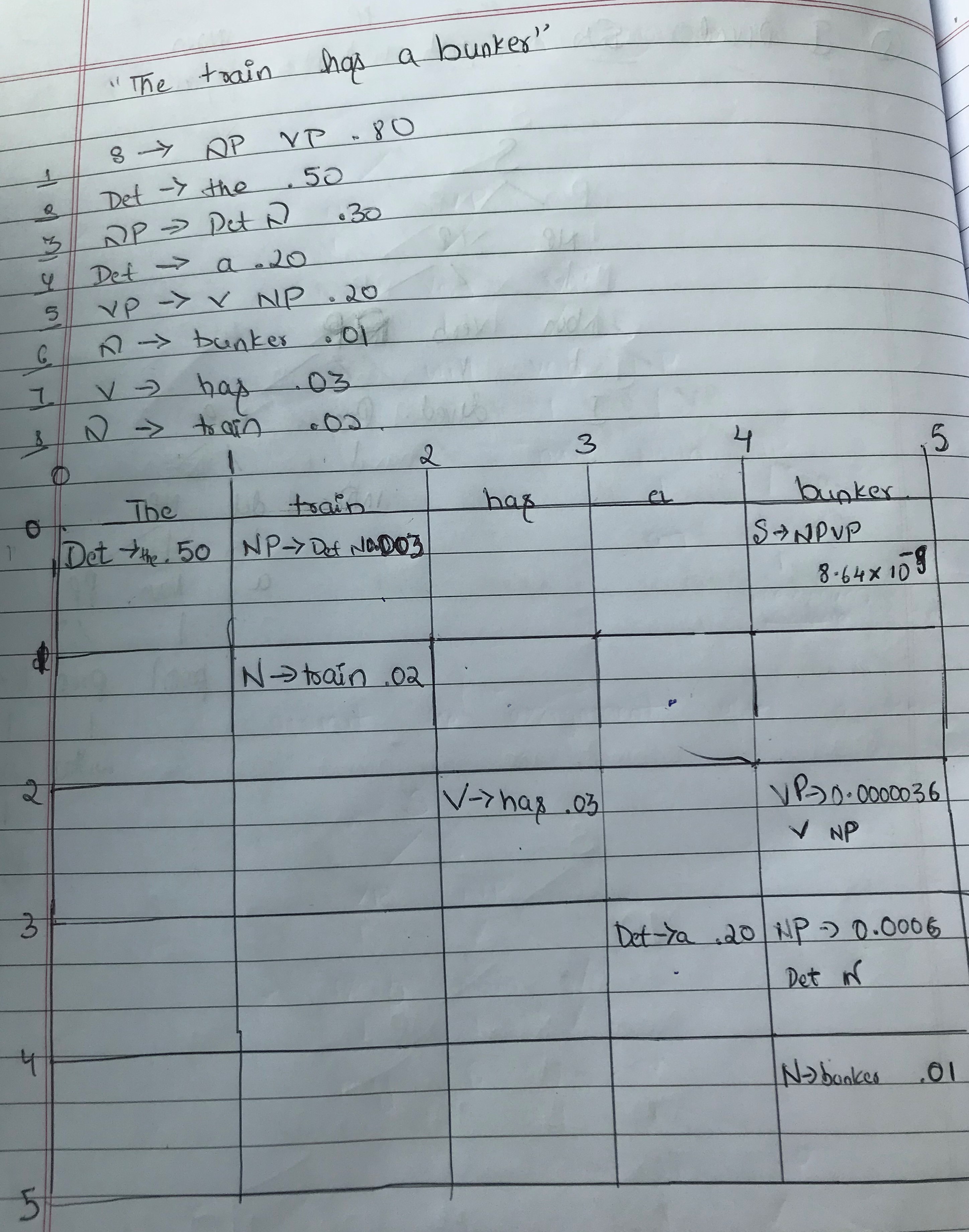
For words not appearing in the corpus or words that are misspelled we can assign a new tag to these class of words.

Further we can use the augmented CKY algorithm to parse the sentences.

Question 3:

Probability of the sentence:

8.64 x



Question 4:

To deal with lexicalized grammar, we must augment the CKY algorithm to accommodate for the change in grammar.

We can use Collins parser to do this.

The first intuition of the Collins parser is to think of the right-hand side of every(internal) CFG rule as consisting of a head non-terminal, together with the nonterminal to the left of the head and the non-terminals to the right of the head. Instead of computing a single probability we can break down the rule via a neat generative story.

We also add a special STOP non-terminal at the left and right edges of the rule. This non-terminal allows the model to know when to stop generating dependents on a given side.

Consider an example:

P(VP(dumped,VBD) -> STOP VBD(dumped, VBD) NP(sacks,NNS) PP(into,P) STOP)

Probability of this rule can be given by :

P(VP(dumped,VBD) -> VBD(dumped, VBD) NP(sacks,NNS) PP(into,P))

If H is a head with head word ‘hw’ and head tag ‘ht’,

lw/lt and rw/rt are the word/tag on the left and right respectively, and P is the parent, then the

probability of an entire rule can be expressed as follows:

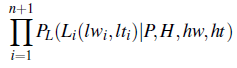
* Step 1:

Generate the head of the phrase H(hw,ht) with probability:



* Step 2:

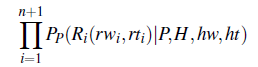
Generate modifiers to the left of the head with total probability



such that Ln+1(lwn+1, ltn+1) =STOP, and we stop generating once we’ve generated a STOP token.

* Step 3.

Generate modifiers to the right of the head with total probability:



such that Rn+1(rwn+1; rtn+1) = STOP, and we stop generating once we’ve generated a STOP token.