Writing a CFG (Context-Free Grammar)

Part 1: Submit a report summary that includes the final grammar that you got with the original sentences in sentences.txt and your two challenge sentences. Then for each sentence,

1. write down the sentence,
2. show the parse tree and
3. write down the rules that you added to camelot_grammar.cfg
4. Add a sentence or two that describes how the rules are applied to parse the sentence, including the significant rules from the grammar; one way is to discuss the parse and describe which part(s) are new for that sentence. You must also explain in your own words how those rules are generating the subphrases. If you had any difficulties, you can describe them.

In [7]:
```python
'''
This small development program can be run to parse the sentences for Homewor
You may either run this as a stand-alone python program or copy it to jupyte
In either case, it must be in the same directory as the sentences and the gr
'''

import nltk

# read the sentences from the file sentences.txt
sentfile = open('sentences.txt', 'r')
# make a list of sentences, separating the tokens by white space.
sentence_list = []
for line in sentfile:
    sentence_list.append(line.split())

# read the grammar file - the nltk data function load will not reload
#    the file unless you set the cache to be False
camg = nltk.data.load('file:camelot_grammar.cfg', cache=False)

# create a recursive descent parser
cam_parser = nltk.RecursiveDescentParser(camg)

# for each sentence print it and its parse trees
# if the grammar cannot parse a sentence, sometimes it gives an error and
#    sometimes it just goes on to the next sentence with no parse tree
for sent in sentence_list:
    print('\n')
    print('Sentence: ')
    print(" ".join(sent))
    print(sent)
    for tree in cam_parser.parse(sent):
        print (tree)
```

```
Sentence:
Arthur is the king .
['Arthur', 'is', 'the', 'king', '.']
(START
  (S1
    (NP (Proper Arthur))
    (VP (VerbT is) (NP (Det the) (NP (Noun king))))
    (Eos .)))


Sentence:
Arthur rides the horse near the castle .
['Arthur', 'rides', 'the', 'horse', 'near', 'the', 'castle', '.']
(START
  (S1
    (NP (Proper Arthur))
    (VP
      (VerbT rides)
      (NP
        (Det the)
        (NP
          (Noun horse)
          (PP (Prep near) (NP (Det the) (NP (Noun castle)))))))
    (Eos .)))
(START
  (S1
    (NP (Proper Arthur))
    (VP
      (VerbT rides)
      (NP (Det the) (NP (Noun horse)))
      (PP (Prep near) (NP (Det the) (NP (Noun castle)))))
    (Eos .)))


Sentence:
Arthur rides the plodding horse near the castle .
['Arthur', 'rides', 'the', 'plodding', 'horse', 'near', 'the', 'castle', '.']
(START
  (S1
    (NP (Proper Arthur))
    (VP
      (VerbT rides)
      (NP
        (Det the)
        (ADJ plodding)
        (NP
          (Noun horse)
          (PP (Prep near) (NP (Det the) (NP (Noun castle)))))))
    (Eos .)))
(START
  (S1
    (NP (Proper Arthur))
    (VP
      (VerbT rides)
      (NP (Det the) (ADJ plodding) (NP (Noun horse)))
      (PP (Prep near) (NP (Det the) (NP (Noun castle)))))
    (Eos .)))


Sentence:
the Holy_Grail is a chalice .
['the', 'Holy_Grail', 'is', 'a', 'chalice', '.']
(START
  (S1
    (NP (Det the) (NP (PNP Holy_Grail)))
    (VP (VerbT is) (NP (Det a) (NP (Noun chalice))))
    (Eos .)))


Sentence:
the sensational Holy_Grail is a sacred chalice .
['the', 'sensational', 'Holy_Grail', 'is', 'a', 'sacred', 'chalice', '.']
(START
  (S1
    (NP (Det the) (ADJ sensational) (NP (PNP Holy_Grail)))
    (VP (VerbT is) (NP (Det a) (ADJ sacred) (NP (Noun chalice))))
    (Eos .)))


Sentence:
every coconut was carried to the hottest mountains .
['every', 'coconut', 'was', 'carried', 'to', 'the', 'hottest', 'mountains', '
.']
(START
  (S1
    (NP (Det every) (NP (Noun coconut)))
    (VP
      (VBD was)
      (VBD carried)
      (PP (TO to) (NP (Det the) (sADJ hottest) (NP (PNS mountains)))))
    (Eos .)))


Sentence:
sixty strangers are at the Round_Table .
['sixty', 'strangers', 'are', 'at', 'the', 'Round_Table', '.']
(START
  (S1
    (NP (CD sixty) (NP (PNS strangers)))
    (VP
      (VBP are)
      (PP (Prep at) (NP (Det the) (NP (PNP Round_Table)))))
    (Eos .)))


Sentence:
Sir_Lancelot might have spoken .
['Sir_Lancelot', 'might', 'have', 'spoken', '.']
(START
  (S1
    (NP (Proper Sir_Lancelot))
    (VP (MD might) (VB have) (VBN spoken))
    (Eos .)))
```

```
Sentence:
Guinevere had been riding with Patsy for five weary nights .
['Guinevere', 'had', 'been', 'riding', 'with', 'Patsy', 'for', 'five', 'weary
', 'nights', '.']
(START
  (S1
    (NP (Proper Guinevere))
    (VP
      (VBD had)
      (VBN been)
      (VBG riding)
      (PP
        (Prep with)
        (NP
          (Proper Patsy)
          (PP (Prep for) (NP (CD five)))
          (ADJ weary)
          (NP (PNS nights)))))
    (Eos .)))


Sentence:
Sir_Bedevere might have been suggesting this quest .
['Sir_Bedevere', 'might', 'have', 'been', 'suggesting', 'this', 'quest', '.']
(START
  (S1
    (NP (Proper Sir_Bedevere))
    (VP
      (MD might)
      (VB have)
      (VBN been)
      (VBG suggesting)
      (NP (Det this) (NP (Noun quest))))
    (Eos .)))


Sentence:
the Britons migrate south frequently .
['the', 'Britons', 'migrate', 'south', 'frequently', '.']
(START
  (S1
    (NP (Det the) (NP (PPNS Britons)))
    (VP (VB migrate) (RB south) (RB frequently))
    (Eos .)))


Sentence:
Arthur and Guinevere ride frequently near the castle .
['Arthur', 'and', 'Guinevere', 'ride', 'frequently', 'near', 'the', 'castle',
'.']
```

```
(START
  (S1
    (NP (Proper Arthur) (CC and) (Proper Guinevere))
    (VP
      (VB ride)
      (RB frequently)
      (PP (Prep near) (NP (Det the) (NP (Noun castle)))))
    (Eos .)))


Sentence:
he suggests to grow fruit at home .
['he', 'suggests', 'to', 'grow', 'fruit', 'at', 'home', '.']
(START
  (S1
    (NP (PRP he))
    (VP
      (VBZ suggests)
      (TO to)
      (VB grow)
      (NP (Noun fruit) (PP (Prep at) (NP (Noun home)))))
    (Eos .)))


Sentence:
riding to Camelot is not hard .
['riding', 'to', 'Camelot', 'is', 'not', 'hard', '.']
(START
  (S1
    (VP (VBG riding) (PP (TO to) (NP (PNP Camelot))))
    (VP (VerbT is) (NOT not) (ADJ hard))
    (Eos .)))


Sentence:
do coconuts speak ?
['do', 'coconuts', 'speak', '?']
(START (S1 (NP (DO do) (NP (PNS coconuts))) (VP (VB speak)) (Eos ?)))


Sentence:
why does England have a king ?
['why', 'does', 'England', 'have', 'a', 'king', '?']
(START
  (S1
    (WAdv why)
    (NP (DO does) (NP (PNP England)))
    (VP (VB have) (NP (Det a) (NP (Noun king))))
    (Eos ?)))
```

Two New Sentences from Challenge.txt

In [18]:
```python
import nltk

# read the sentences from the file sentences.txt
sentfile = open('challenge.sentences.txt', 'r')
# make a list of sentences, separating the tokens by white space.
sentence_list = []
for line in sentfile:
    sentence_list.append(line.split())

# read the grammar file - the nltk data function load will not reload
#    the file unless you set the cache to be False
camg = nltk.data.load('file:camelot_grammar.cfg', cache=False)

# create a recursive descent parser
cam_parser = nltk.RecursiveDescentParser(camg)

# for each sentence print it and its parse trees
# if the grammar cannot parse a sentence, sometimes it gives an error and
#    sometimes it just goes on to the next sentence with no parse tree
i =0
for sent in sentence_list:

    i=i+1
    if(i!=4 and i!=7): continue
    print('\n')
    print('Sentence: ')
    print(" ".join(sent))
    print(sent)
    for tree in cam_parser.parse(sent):
        print (tree)
```

```
Sentence:
it is Sir_Lancelot who knows Zoot !
['it', 'is', 'Sir_Lancelot', 'who', 'knows', 'Zoot', '!']
(START
  (S1
    (NP (PRP it))
    (VP (VerbT is) (NP (Proper Sir_Lancelot)))
    (NP (WPro who) (VP (VBZ knows) (NP (Proper Zoot))))
    (Eos !)))


Sentence:
Arthur rode to Camelot and drank from his chalice .
['Arthur', 'rode', 'to', 'Camelot', 'and', 'drank', 'from', 'his', 'chalice',
'.']
(START
  (S1
    (NP (Proper Arthur))
    (VP (VBD rode) (PP (TO to) (NP (PNP Camelot))))
    (CC and)
    (VP (VBD drank) (PP (Prep from) (NP (POS his) (Noun chalice))))
    (Eos .)))
```

Understanding the Grammar Rules

Grammar rules for the first sentence:

S1 -> NP VP NP Eos NP -> PRP VP -> VerbT NP NP -> Proper NP -> WPro VP NP VP -> VBZ NP NP -> Proper Grammar rules for the second sentence:

S1 -> NP VP CC VP Eos NP -> Proper VP -> VBD PP PP -> TO NP NP -> PNP CC -> and VP -> VBD PP PP -> Prep NP NP -> POS Noun

Here's a more detailed explanation of the grammar rules for the given sentences:

For the first sentence:

- S1 -> NP VP NP Eos: This rule represents the structure of a sentence (S1) consisting of a noun phrase (NP), a verb phrase (VP), another noun phrase (NP), and an end-of-sentence marker (Eos).
- NP -> PRP: This rule indicates that a noun phrase (NP) can be a pronoun (PRP).
- VP -> VerbT NP: This rule states that a verb phrase (VP) consists of a transitive verb (VerbT) followed by a noun phrase (NP).

- NP -> Proper: This rule shows that a noun phrase (NP) can be a proper noun (Proper).
- NP -> WPro VP NP: This rule represents a noun phrase (NP) that consists of a wh-pronoun (WPro), followed by a verb phrase (VP), and another noun phrase (NP).
- VP -> VBZ NP: This rule indicates that a verb phrase (VP) can be a present tense verb (VBZ) followed by a noun phrase (NP).

Overall, these grammar rules capture the structure of the first sentence, involving noun phrases, verb phrases, pronouns, and proper nouns.

For the second sentence:

- S1 -> NP VP CC VP Eos: This rule represents the structure of a sentence (S1) consisting of a noun phrase (NP), a verb phrase (VP), a coordinating conjunction (CC), another verb phrase (VP), and an end-of-sentence marker (Eos).
- NP -> Proper: This rule indicates that a noun phrase (NP) can be a proper noun (Proper).
- VP -> VBD PP: This rule shows that a verb phrase (VP) consists of a past tense verb (VBD) followed by a prepositional phrase (PP).
- PP -> TO NP: This rule represents a prepositional phrase (PP) consisting of the preposition "to" (TO) followed by a noun phrase (NP).
- NP -> PNP: This rule indicates that a noun phrase (NP) can be a proper noun phrase (PNP).
- CC -> and: This rule simply represents the coordinating conjunction "and".
- VP -> VBD PP: This rule indicates that a verb phrase (VP) can be a past tense verb (VBD) followed by a prepositional phrase (PP).
- PP -> Prep NP: This rule represents a prepositional phrase (PP) consisting of a preposition (Prep) followed by a noun phrase (NP).
- NP -> POS Noun: This rule indicates that a noun phrase (NP) can be a possessive pronoun (POS) followed by a noun (Noun).

These grammar rules capture the structure of the second sentence, involving noun phrases, verb phrases, prepositional phrases, coordinating conjunctions, and various parts of speech.

Part-2

Give your two exemplar “sentences”. For the first sentence, explain what part of the phrase structure is not represented by the grammar in part 1 and show the parser output with no parse For the second “sentence”, explain how overgeneralization allows this so-called sentence to be parsed and give the parse tree output from the parser

In [34]:
```
'''
PART 2 A
'''

import nltk

# read the sentences from the file sentences.txt
sentfile = open('NewSentencesOne.txt', 'r')
# make a list of sentences, separating the tokens by white space.
sentence_list = []
for line in sentfile:
    sentence_list.append(line.split())

# read the grammar file - the nltk data function load will not reload
#    the file unless you set the cache to be False
camg = nltk.data.load('file:camelot_grammar.cfg', cache=False)

# create a recursive descent parser
cam_parser = nltk.RecursiveDescentParser(camg)

# for each sentence print it and its parse trees
# if the grammar cannot parse a sentence, sometimes it gives an error and
#    sometimes it just goes on to the next sentence with no parse tree
for sent in sentence_list:
    print('\n')
    print('Sentence: ')
    print(" ".join(sent))
    print(sent)
    for tree in cam_parser.parse(sent):
        print (tree)
```

```
Sentence:
Arthur drinks at the castles .
['Arthur', 'drinks', 'at', 'the', 'castles', '.']
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[34], line 31
     29 print(" ".join(sent))
     30 print(sent)
---> 31 for tree in cam_parser.parse(sent):
     32     print (tree)

File /opt/homebrew/Cellar/jupyterlab/4.0.0/libexec/lib/python3.11/site-packag
es/nltk/parse/recursivedescent.py:76, in RecursiveDescentParser.parse(self, t
okens)
     72 def parse(self, tokens):
     73     # Inherit docs from ParserI
     75     tokens = list(tokens)
---> 76     self._grammar.check_coverage(tokens)
     78     # Start a recursive descent parse, with an initial tree
     79     # containing just the start symbol.
     80     start = self._grammar.start().symbol()

File /opt/homebrew/Cellar/jupyterlab/4.0.0/libexec/lib/python3.11/site-packag
es/nltk/grammar.py:665, in CFG.check_coverage(self, tokens)
    663 if missing:
    664     missing = ", ".join(f"{w!r}" for w in missing)
--> 665     raise ValueError(
    666         "Grammar does not cover some of the " "input words: %r." %
missing
    667     )

ValueError: Grammar does not cover some of the input words: "'castles'".
```

Explaination for Part 2 a Error :

As we can see we get an error since "castles" is not defined in our grammar. But below you can see for 2b the changes I have made and see the correct output

In [33]:
```python
'''
PART 2 B
'''

import nltk

# read the sentences from the file sentences.txt
sentfile = open('NewSentencesTwo.txt', 'r')
# make a list of sentences, separating the tokens by white space.
sentence_list = []
for line in sentfile:
    sentence_list.append(line.split())

# read the grammar file - the nltk data function load will not reload
#    the file unless you set the cache to be False
camg = nltk.data.load('file:camelot_grammar.cfg', cache=False)

# create a recursive descent parser
cam_parser = nltk.RecursiveDescentParser(camg)

# for each sentence print it and its parse trees
# if the grammar cannot parse a sentence, sometimes it gives an error and
#    sometimes it just goes on to the next sentence with no parse tree
for sent in sentence_list:
    print('\n')
    print('Sentence: ')
    print(" ".join(sent))
    print(sent)
    for tree in cam_parser.parse(sent):
        print (tree)
```

```
Sentence:
Arthur drinks at the castle .
['Arthur', 'drinks', 'at', 'the', 'castle', '.']
(START
  (S1
    (NP (Proper Arthur))
    (VP
      (VerbT drinks)
      (PP (Prep at) (NP (Det the) (NP (Noun castle)))))
    (Eos .)))


Sentence:
riding to England is not hard .
['riding', 'to', 'England', 'is', 'not', 'hard', '.']
(START
  (S1
    (VP (VBG riding) (PP (TO to) (NP (PNP England))))
    (VP (VerbT is) (NOT not) (ADJ hard))
    (Eos .)))
```

Explaination for the Part 2B

For these two sentences we have grammar already defined in the part 1. So hence we are able to parse and form the tree.

Sentence: Arthur drinks at the castle. Sentence Structure: S1 NP (Noun Phrase): Arthur (Proper noun) VP (Verb Phrase): drinks PP (Prepositional Phrase): at the castle Preposition: at NP: the castle Determiner: the Noun: castle Eos: Sentence-ending punctuation (.)

Sentence: Riding to England is not hard. Sentence Structure: S1 VP: riding to England Verb: riding (gerund form) PP: to England Preposition: to NP: England VP: is not hard Verb: is NOT: not ADJ: hard Eos: Sentence-ending punctuation (.) In both sentences, the structure follows a basic pattern of subject-verb-object (SVO) or subject-verb-complement (SVC). The specific grammatical constituents are labeled accordingly, including noun phrases (NP), verb phrases (VP), prepositional phrases (PP), determiners (Det), nouns (Noun), adjectives (ADJ), and sentence-ending punctuation (Eos).

A more detailed explanation of the grammar rules for the above sentences:

1. Sentence: Arthur drinks at the castle.

- The sentence structure is labeled as S1, indicating a basic declarative sentence.
- NP (Noun Phrase): "Arthur" is the subject of the sentence. It is labeled as a proper noun (Proper) since it represents a specific person.
- VP (Verb Phrase): "drinks" is the main verb of the sentence. It describes the action performed by the subject.
- PP (Prepositional Phrase): "at the castle" is a phrase that provides additional information about the location of the action.
  - Preposition: "at" introduces the prepositional phrase and indicates the relationship between the verb and the noun phrase that follows.
  - NP: "the castle" is the object of the preposition. It consists of a determiner "the" followed by a noun "castle" that specifies the location.
- Eos: "Eos" represents the sentence-ending punctuation, which in this case is a period (.).

2. Sentence: Riding to England is not hard.

- The sentence structure is again labeled as S1, indicating a declarative sentence.
- VP: "riding to England" is the main verb phrase. It describes the action being performed.
  - Verb: "riding" is a gerund form of the verb, indicating an ongoing action.
  - PP: "to England" is a prepositional phrase that provides information about the destination of the action.
    - Preposition: "to" introduces the prepositional phrase and indicates the direction or goal of the action.
    - NP: "England" is the object of the preposition, representing the specific destination.
- VP: "is not hard" is another verb phrase that provides additional information about the action.
  - Verb: "is" serves as the linking verb, connecting the subject with the complement.
  - NOT: "not" is a negation marker indicating the action is negated.
  - ADJ: "hard" is an adjective that describes the difficulty level of the action.
- Eos: The sentence-ending punctuation is a period (.), indicating the end of the sentence.

By breaking down the sentences into their constituent grammatical elements, we can understand the syntactic structure and relationships between the different parts of speech. This analysis helps in comprehending the meaning and syntax of the sentences.

```
In [ ]:
```

In [ ]:

```
```

In [ ]:

```
```