LAB-12 Building and Parsing Context Free Grammars

ROLLNO : 225229110

NAME : Hari Prasath

In [1]:
```
import nltk
nltk.download("punkt")
from nltk.tree import Tree
from nltk.tokenize import word_tokenize
from IPython.display import display
import nltk,re,pprint
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
import numpy as npt
!apt-get install -y xvfb # Install X Virtual Frame Buffer
import os
os.system('Xvfb :1 -screen 0 1600x1200x16 &')# create virtual display with siz
os.environ['DISPLAY']=':1.0'# tell X clients to use our virtual DISPLAY :1.0.
%matplotlib inline
### INSTALL GHOSTSCRIPT (Required to display NLTK trees)
!apt install ghostscript python3-tk
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ashac\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
'apt-get' is not recognized as an internal or external command,
operable program or batch file.
'apt' is not recognized as an internal or external command,
operable program or batch file.
```

EXERCISE-1: Build Grammar and Parser

In [ ]:
```
rammar_1 = nltk.CFG.fromstring("""
S -> NP VP | NP VP
NP -> N | Det N | PRO | N N
VP -> V NP CP | VP ADVP | V NP
ADVP -> ADV ADV
CP -> COMP S
N -> 'Lisa' | 'brother' | 'peanut' | 'butter'
V -> 'told' | 'liked'
COMP -> 'that'
Det -> 'her'
PRO -> 'she'
ADV -> 'very' | 'much'
S -> NP VP
NP -> NP CONJ NP | N | NP PP | Det N | N | Det N
VP -> VP PP | VP CONJ VP | V | V
PP -> P NP | P NP
N -> 'Homer' | 'friends' | 'work' | 'bar'
V -> 'drank' | 'sang'
CONJ -> 'and' | 'and'
Det -> 'his' | 'the'
P -> 'from' | 'in'
S -> NP VP
NP -> NP CONJ NP | N | N
VP -> V ADJP
ADJP -> ADJP CONJ ADJP | ADJ | ADV ADJ
N -> 'Homer' | 'Marge'
V -> 'are'
CONJ -> 'and' | 'but'
ADJ -> 'poor' | 'happy'
ADV -> 'very'
S -> NP VP | NP AUX VP
NP -> PRO | NP CP | Det N | PRO | PRO | PRO | N |Det N
VP -> V NP PP | V NP NP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he' | 'I' | 'him'
N -> 'book' | 't' | 'sister'
V -> 'gave' | 'given'
COMP -> 'that'
AUX -> 'had'
P -> 'to'
S -> NP VP
NP -> PRO | Det N | Det N
VP -> V NP PP
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he'
N -> 'book' | 'sister'
V -> 'gave'
P -> 'to'
S -> NP VP
NP -> Det ADJ N | Det ADJ ADJ N | N
VP -> V NP|VP PP
PP -> P NP
Det -> 'the' | 'the'
ADJ -> 'big' | 'tiny' | 'nerdy'
N -> 'bully' | 'kid' | 'school'
V -> 'punched'
P -> 'after'
""")
```

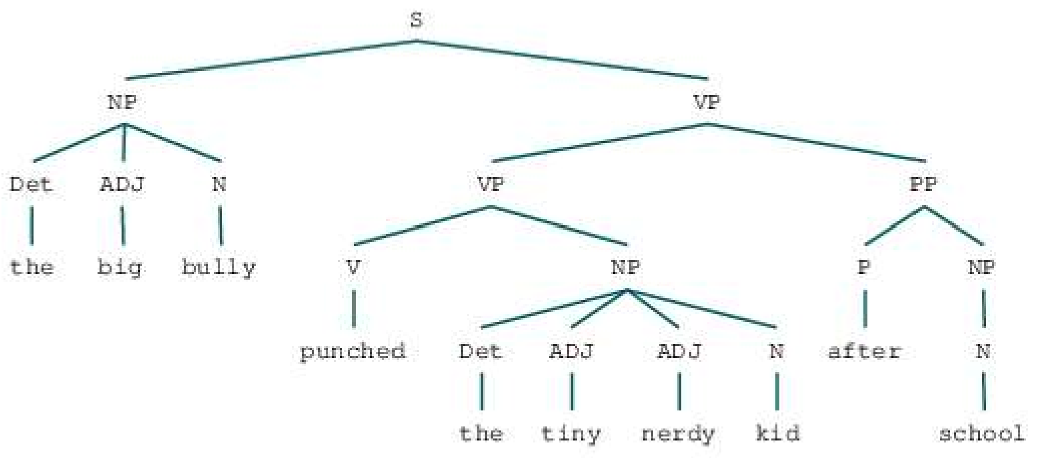
1.Using NLTK's nltk.CFG.fromstring() method, build a CFG named grammar1. The grammar should cover all of the sentences below and their tree structure as presented on this page. The grammar's start symbol should be 'S': make sure that an S rule (ex. S -> NP VP) is the very top rule in your list of rules. (s6)the big bully punched the tiny nerdy kid after school

In [ ]:
```
s6_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> Det ADJ N | Det ADJ ADJ N | N
VP -> V NP|VP PP
PP -> P NP
Det -> 'the' | 'the'
ADJ -> 'big' | 'tiny' | 'nerdy'
N -> 'bully' | 'kid' | 'school'
V -> 'punched'
P -> 'after'
""")
```

In [ ]:
```
sent1 = word_tokenize("the big bully punched the tiny nerdy kid after school")
parser = nltk.ChartParser(s6_grammar1)
for tree in parser.parse(sent1):
 print(tree)
```

```
(S
  (NP (Det the) (ADJ big) (N bully))
  (VP
    (VP (V punched) (NP (Det the) (ADJ tiny) (ADJ nerdy) (N kid)))
    (PP (P after) (NP (N school)))))
```

In [ ]:
```
np1 =nltk.Tree.fromstring('(S(NP (Det the) (ADJ big) (N bully))(VP(VP (V punch
display(np1)
```



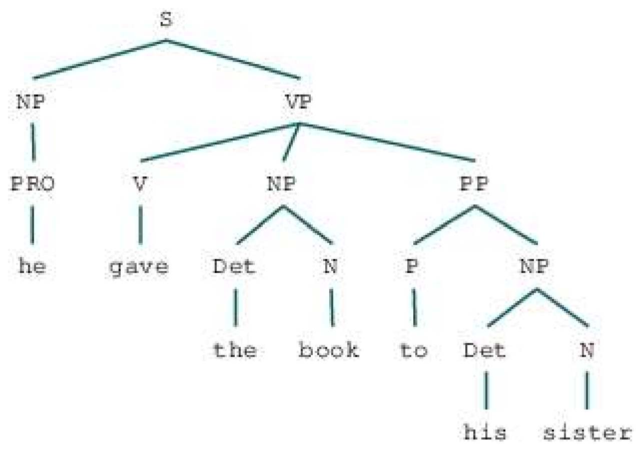
(s7)he gave the book to his sister

In [ ]:
```
s7_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> PRO | Det N | Det N
VP -> V NP PP
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he'
N -> 'book' | 'sister'
V -> 'gave'
P -> 'to'
""")
```

In [ ]:
```
sent2 = word_tokenize("he gave the book to his sister")
parser = nltk.ChartParser(s7_grammar1)
for i in parser.parse(sent2):
  print(i)
```

```
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP (Det the) (N book))
    (PP (P to) (NP (Det his) (N sister)))))
```

In [ ]:
```
np2 =nltk.Tree.fromstring('(S(NP (PRO he))(VP(V gave)(NP (Det the) (N book))(P
display(np2)
```



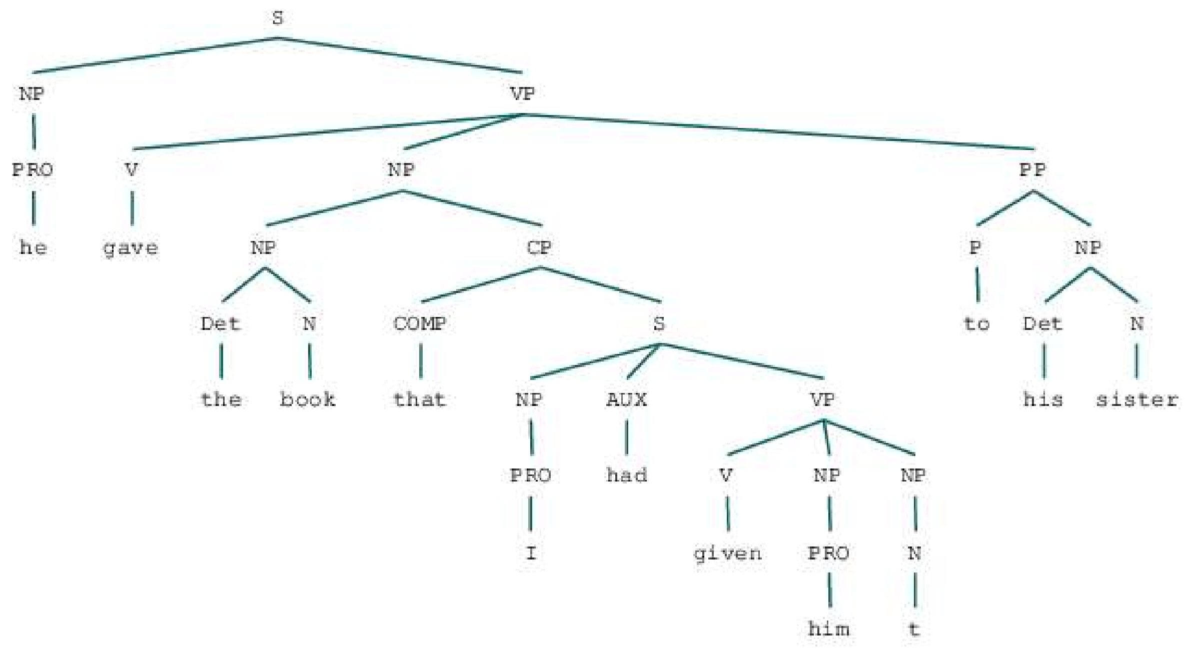
(s8)he gave the book that I had given him t to his sister

In [ ]:
```
s8_grammar1 = nltk.CFG.fromstring("""
S -> NP VP | NP AUX VP
NP -> PRO | NP CP | Det N | PRO | PRO | PRO | N |Det N
VP -> V NP PP | V NP NP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his'
PRO -> 'he' | 'I' | 'him'
N -> 'book' | 't' | 'sister'
V -> 'gave' | 'given'
COMP -> 'that'
AUX -> 'had'
P -> 'to'
""")
```

In [ ]:
```
sent3 = word_tokenize("he gave the book that I had given him t to his sister")
parser = nltk.ChartParser(s8_grammar1)
for i in parser.parse(sent3):
  print(i)
```

```
(S
  (NP (PRO he))
  (VP
    (V gave)
    (NP
      (NP (Det the) (N book))
      (CP
        (COMP that)
        (S
          (NP (PRO I))
          (AUX had)
          (VP (V given) (NP (PRO him)) (NP (N t))))))
    (PP (P to) (NP (Det his) (N sister)))))
```

In [ ]:
```
np3 =nltk.Tree.fromstring('(S(NP (PRO he))(VP(V gave)(NP(NP (Det the) (N book)
display(np3)
```



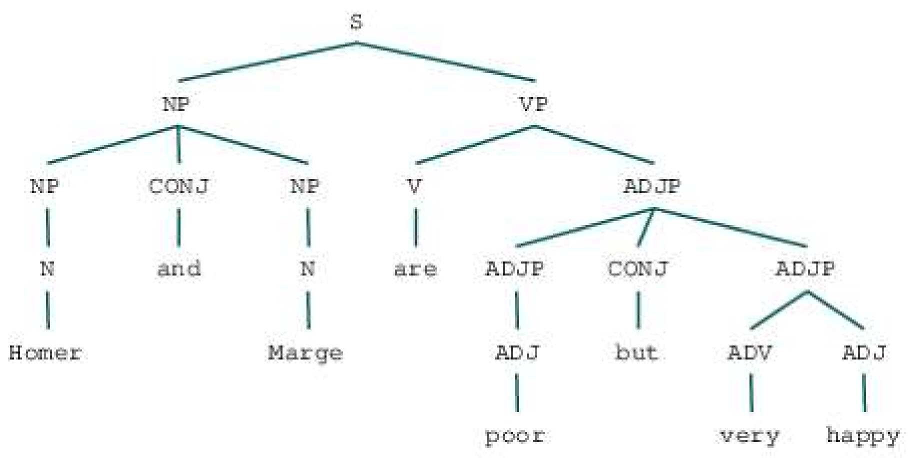
(s9)Homer and Marge are poor but very happy

In [ ]:
```
s9_grammar1 = nltk.CFG.fromstring("""
S -> NP VP
NP -> NP CONJ NP | N | N
VP -> V ADJP
ADJP -> ADJP CONJ ADJP | ADJ | ADV ADJ
N -> 'Homer' | 'Marge'
V -> 'are'
CONJ -> 'and' | 'but'
ADJ -> 'poor' | 'happy'
ADV -> 'very'
""")
```

In [ ]:
```
sent4 = word_tokenize("Homer and Marge are poor but very happy")
parser = nltk.ChartParser(s9_grammar1)
for i in parser.parse(sent4):
  print(i)
```

```
(S
  (NP (NP (N Homer)) (CONJ and) (NP (N Marge)))
  (VP
    (V are)
    (ADJP (ADJP (ADJ poor)) (CONJ but) (ADJP (ADV very) (ADJ happy)))))
```

```
In [ ]: np4 =nltk.Tree.fromstring('(S(NP (NP (N Homer)) (CONJ and) (NP (N Marge)))(VP(
        display(np4)
```



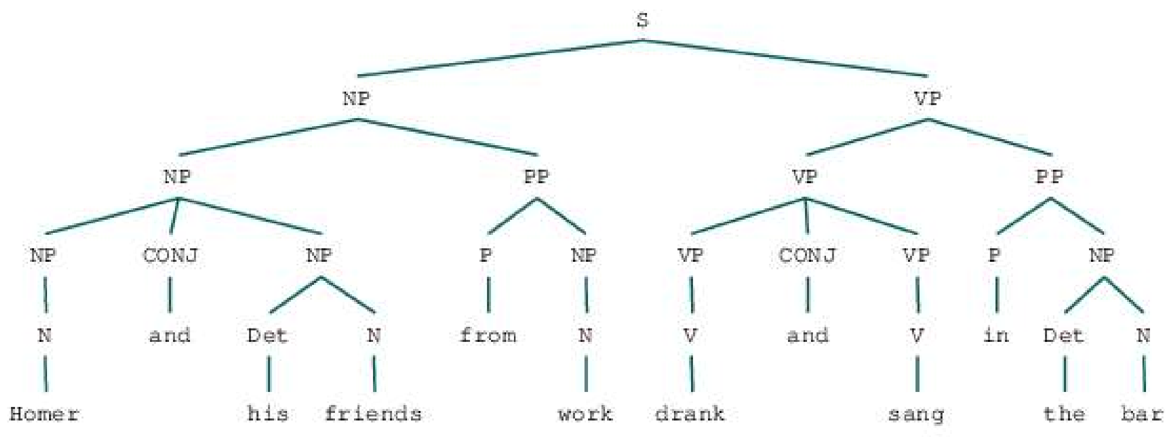
(s10)Homer and his friends from work drank and sang in the bar

```
In [ ]: s10_grammar1 = nltk.CFG.fromstring("""
        S -> NP VP
        NP -> NP CONJ NP | N | NP PP | Det N | N | Det N
        VP -> VP PP | VP CONJ VP | V | V
        PP -> P NP | P NP
        N -> 'Homer' | 'friends' | 'work' | 'bar'
        V -> 'drank' | 'sang'
        CONJ -> 'and' | 'and'
        Det -> 'his' | 'the'
        P -> 'from' | 'in'
        """)
```

In [ ]:
```
sent5 = word_tokenize("Homer and his friends from work drank and sang in the b
parser = nltk.ChartParser(s10_grammar1)
for i in parser.parse(sent5):
  print(i)
```

```
(S
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
  (VP
    (VP (VP (V drank)) (CONJ and) (VP (V sang)))
    (PP (P in) (NP (Det the) (N bar)))))
(S
  (NP
    (NP (NP (N Homer)) (CONJ and) (NP (Det his) (N friends)))
    (PP (P from) (NP (N work))))
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
(S
  (NP
    (NP (N Homer))
    (CONJ and)
    (NP (NP (Det his) (N friends)) (PP (P from) (NP (N work)))))
  (VP
    (VP (V drank))
    (CONJ and)
    (VP (VP (V sang)) (PP (P in) (NP (Det the) (N bar))))))
```

In [ ]:
```
np5 =nltk.Tree.fromstring('(S(NP(NP (NP (N Homer)) (CONJ and) (NP (Det his) (N
display(np5)
```



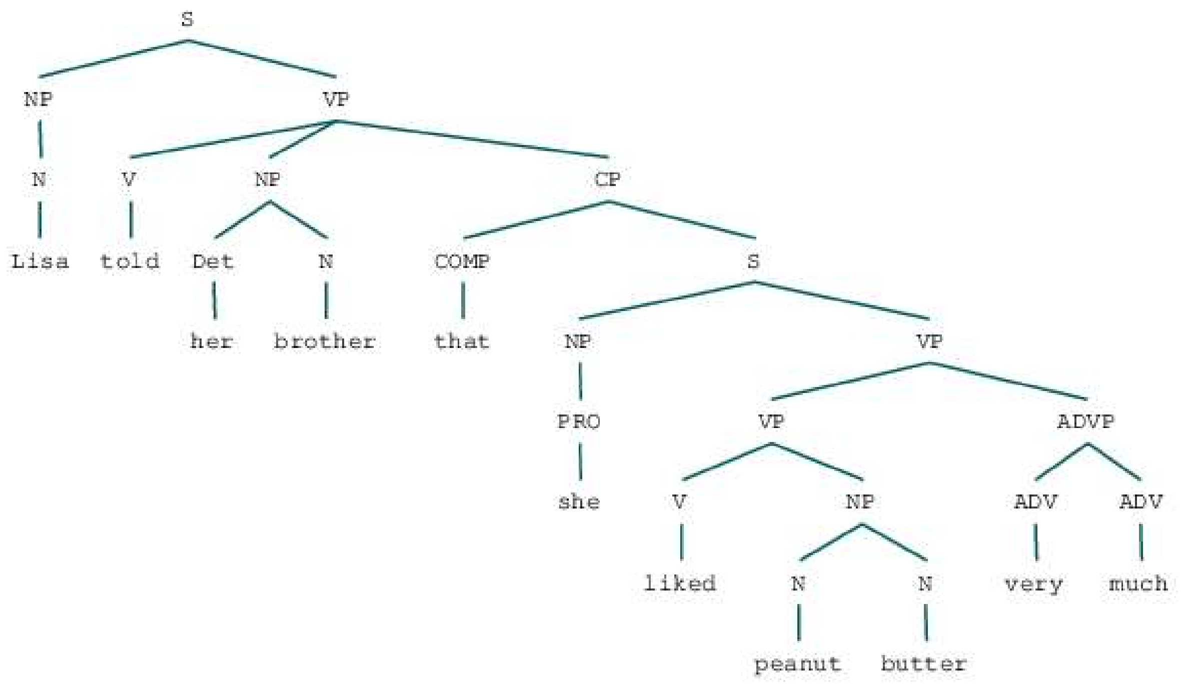
(s11)Lisa told her brother that she liked peanut butter very much

In [ ]:
```
s11_grammar1 = nltk.CFG.fromstring("""
S -> NP VP | NP VP
NP -> N | Det N | PRO | N N
VP -> V NP CP | VP ADVP | V NP
ADVP -> ADV ADV
CP -> COMP S
N -> 'Lisa' | 'brother' | 'peanut' | 'butter'
V -> 'told' | 'liked'
COMP -> 'that'
Det -> 'her'
PRO -> 'she'
ADV -> 'very' | 'much'
""")
```

In [ ]:
```
sent6 = word_tokenize("Lisa told her brother that she liked peanut butter very
parser = nltk.ChartParser(s11_grammar1)
for i in parser.parse(sent6):
  print(i)
```

```
(S
  (NP (N Lisa))
  (VP
    (VP
      (V told)
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S (NP (PRO she)) (VP (V liked) (NP (N peanut) (N butter))))))
    (ADVP (ADV very) (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP (V liked) (NP (N peanut) (N butter)))
          (ADVP (ADV very) (ADV much)))))))
```

In [ ]:
```
np6 =nltk.Tree.fromstring('(S(NP (N Lisa))(VP(V told)(NP (Det her) (N brother)
display(np6)
```



2.Once a grammar is built, you can print it. Also, you can extract a set of production rules with the .productions() method. Unlike the .productions() method called on a Tree object, the resulting list should be duplicate-free. As before, each rule in the list is a production rule type. A rule has a left-hand side node (the parent node), which you can getto using the .lhs() method; the actual string label for the node can be accessed by calling .symbol() on the node object.

In [ ]:
```
grammer3 = nltk.CFG.fromstring("""
S -> NP VP
NP -> N
VP -> V
N -> 'Homer'
V -> 'sleeps'
""")
```

In [ ]:
```
print(grammer3)
```

```
Grammar with 5 productions (start state = S)
    S -> NP VP
    NP -> N
    VP -> V
    N -> 'Homer'
    V -> 'sleeps'
```

In [ ]:
```
grammer3.productions()
```

Out[27]:
```
[S -> NP VP, NP -> N, VP -> V, N -> 'Homer', V -> 'sleeps']
```

In [ ]:
```
last_rule = grammer3.productions()[-1]
last_rule
```

Out[29]:
```
V -> 'sleeps'
```

In [ ]:
```
last_rule.is_lexical()
```

Out[30]:
```
True
```

In [ ]:
```
last_rule.lhs()
```

Out[31]:
```
V
```

In [ ]:
```
last_rule.lhs().symbol()
```

Out[32]:
```
'V'
```

3.Explore the rules and answer the following questions.

In [ ]:
```
Grammar_all = nltk.CFG.fromstring("""
S -> NP VP | NP AUX VP
NP -> Det ADJ N | N | PRO | Det N | PRO | NP CP | PRO | NP CONJ | NP PP | N N
VP -> V NP | VP PP | V NP PP | V NP | V ADJP | VP PP | VP CONJ | V NP CP | VP
CP -> COMP S
PP -> P NP
Det -> 'the' | 'his' | 'her'
ADJ -> 'big' | 'tiny' | 'nerdy' | 'poor' | 'happy'
ADV -> 'very' | 'much'
PRO -> 'he' | 'I' | 'him' | 'she'
ADJP -> ADJP CONJ | ADJ
ADVP -> ADV
N -> 'bully' | 'kid' | 'school' | 'book' | 'sister' | 't' | 'Homer' | 'Marge'|
V -> 'punched' | 'gave' | 'given' | 'are' | 'drank' | 'sang' | 'told' | 'liked
CONJ -> 'and' | 'but'
COMP -> 'that'
AUX -> 'had'
P -> 'after' | 'to' | 'from' | 'in'
""")
```

a. What is the start state of your grammar?

In [ ]:
```
Grammar_all.productions()[0].lhs()
```

Out[38]:
```
S
```

b. How many CF rules are in your grammar?

```
In [ ]: len(Grammar_all.productions())

Out[39]: 71
```

c. How many of them are lexical?

```
In [ ]: n=0
        for x in Grammar_all.productions():
            if x.is_lexical():
                n = n+1
        print("How many of them are lexical? ",n)

        How many of them are lexical?  45
```

d. How many VP rules are there? That is, how many rules have 'VP' on the left-hand side of the rule? That is, how many rules are of the VP -> ... form?

```
In [ ]: n=0
        for x in Grammar_all.productions():
            if x.lhs().symbol() == 'VP':
                n = n+1
        n

Out[42]: 9
```

e. How many V rules are there? That is, how many rules have 'V' on the left-hand side of the fule? That is, how many rules are of the V -> ... form?

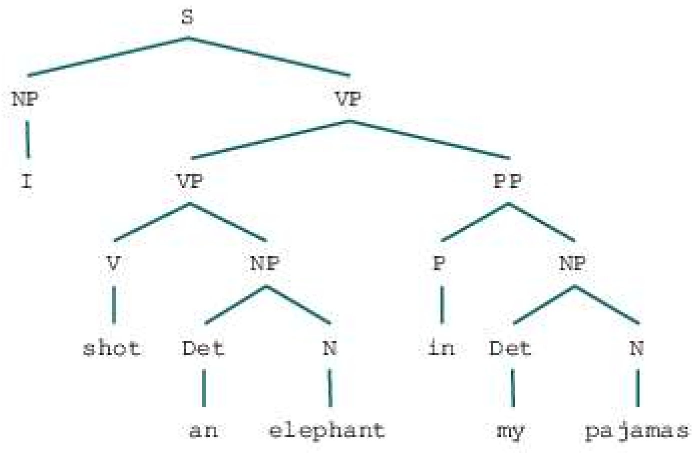
```
In [ ]: n=0
        for x in Grammar_all.productions():
            if x.lhs().symbol() == 'V':
                n = n+1
        n

Out[43]: 8
```

4.Using grammar1, build a chart parser.

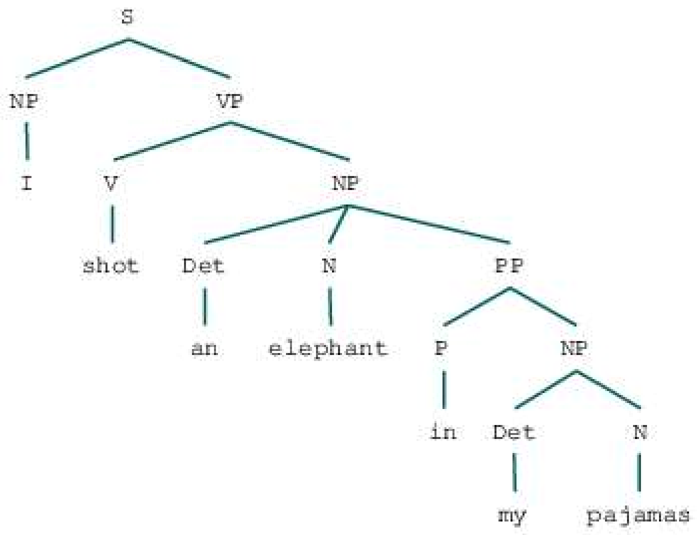
In [ ]:
```
sent = word_tokenize("Lisa told her brother that she liked peanut butter very
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent):
    print(i)
```

```
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP (Det her) (N brother))
    (CP
      (COMP that)
      (S
        (NP (PRO she))
        (VP
          (VP
            (VP (V liked) (NP (N peanut) (N butter)))
            (ADVP (ADV very)))
          (ADVP (ADV much)))))))
(S
  (NP (N Lisa))
  (VP
    (V told)
    (NP
      (NP (Det her) (N brother))
      (CP
        (COMP that)
        (S
          (NP (PRO she))
          (VP
            (VP
              (VP (V liked) (NP (N peanut) (N butter)))
              (ADVP (ADV very)))
            (ADVP (ADV much))))))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (VP
        (V told)
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP (V liked) (NP (N peanut) (N butter))))))
      (ADVP (ADV very)))
    (ADVP (ADV much))))
(S
  (NP (N Lisa))
  (VP
    (VP
      (VP
        (V told)
        (NP
          (NP (Det her) (N brother))
          (CP
            (COMP that)
            (S
              (NP (PRO she))
              (VP (V liked) (NP (N peanut) (N butter)))))))
      (ADVP (ADV very)))
      (ADVP (ADV much))))
  (S
    (NP (N Lisa))
    (VP
      (VP
        (V told)
        (NP (Det her) (N brother))
        (CP
          (COMP that)
          (S
            (NP (PRO she))
            (VP
              (VP (V liked) (NP (N peanut) (N butter)))
              (ADVP (ADV very))))))
      (ADVP (ADV much))))
  (S
    (NP (N Lisa))
    (VP
      (VP
        (V told)
        (NP
          (NP (Det her) (N brother))
          (CP
            (COMP that)
            (S
              (NP (PRO she))
              (VP
                (VP (V liked) (NP (N peanut) (N butter)))
                (ADVP (ADV very)))))))
      (ADVP (ADV much))))
```

In [50]:
```
q41 =nltk.Tree.fromstring('(S (NP I) (VP (VP (V shot) (NP (Det an) (N elephant
display(q41)
```



In [51]:
```
q42 =nltk.Tree.fromstring('(S (NP I) (VP (V shot) (NP (Det an) (N elephant) (P
display(q42)
```



5. Using the parser, parse the sentences s6 -- s11. If your grammar1 is built correctly to cover all of the sentences, the parser should successfully parse all of them.

In [52]:
```
!pip install simple-colors
from simple_colors import *
```

```
Collecting simple-colors
  Downloading simple_colors-0.1.5-py3-none-any.whl (2.8 kB)
Installing collected packages: simple-colors
Successfully installed simple-colors-0.1.5
```

In [54]:
```
print(black("(s6):the big bully punched the tiny nerdy kid after school","bold
print("\n")
sent6 = word_tokenize("the big bully punched the tiny nerdy kid after school")
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent6):
    print(i)
print("-------------------------------------------------------------------------
print("\n")
print(black("(s7):he gave the book to his sister","bold"))
print("\n")
sent7 = word_tokenize("he gave the book to his sister")
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent7):
    print(i)
print("-------------------------------------------------------------------------
print("\n")
print(black("(s8):he gave the book that I had given him t to his sister","bold
print("\n")
sent8 = word_tokenize("he gave the book that I had given him t to his sister")
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent8):
    print(i)
print("-------------------------------------------------------------------------
print("\n")
print(black("(s9):Homer and Marge are poor but very happy","bold"))
print("\n")
sent9 = word_tokenize("Homer and Marge are poor but very happy")
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent9):
    print(i)
print("-------------------------------------------------------------------------
print("\n")
print(black("(s10):Homer and his friends from work drank and sang in the bar",
print("\n")
sent10 = word_tokenize("Homer and his friends from work drank and sang in the
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent10):
    print(i)
print("-------------------------------------------------------------------------
print("\n")
print(black("(s11):Lisa told her brother that she liked peanut butter very muc
print("\n")
sent11 = word_tokenize("Lisa told her brother that she liked peanut butter ver
parser = nltk.ChartParser(Grammar_all)
for i in parser.parse(sent11):
    print(i)
```

```
(s6):the big bully punched the tiny nerdy kid after school


--------------------------------------------------------------------------
----


(s7):he gave the book to his sister


(S
  (NP (PRO he))
  (VP
    (VP (V gave) (NP (Det the) (N book)))
    (PP (P to) (NP (Det his) (N sister)))))
(S
  (NP (PRO he))
  (VP
    (V gave)
```

In [ ]: