# Labsheet 11: Building Parse Trees

## Name: Ezhilarasan C

## Roll No:225229151

### Exercise 1

In [1]:
```
import nltk,re,pprint
from nltk.tree import Tree
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.chunk import ne_chunk
import numpy as npt
```

In [2]:
```
np= nltk.Tree.fromstring('(NP (N Marge))')
np.pretty_print()
```

```
  NP
  |
  N
  |
Marge
```

In [3]:
```
ec= nltk.Tree.fromstring('(VP (V make) (NP (DET a) (N ham) (N sandwich)))')
ec.pretty_print()
```

```
          VP
  ________|___
 |            NP
 |     _______|_____
 V   DET      N     N
 |    |       |     |
make  a      ham sandwich
```

In [4]:
```
au= nltk.Tree.fromstring('(AUX will)')
au.pretty_print()
```

```
AUX
 |
will
```

### Excercise-2

In [5]:

```
tree = nltk.Tree.fromstring('(NP (Adj old) (NP (N men) (Conj and) (N women)))')
tree.pretty_print()
```

```
         NP
  _______|___
 |           NP
 |    _______|_____
Adj  N      Conj  N
 |   |       |     |
old men     and  women
```

### Exercise- 3

In [17]:

```
s= nltk.Tree.fromstring('(S (NP (N Marge)) (AUX will) (VP (V make) (NP (DET a) (N
s.pretty_print()
```

```
            S
   _________|________
  |    |             VP
  |    |     ________|___
  NP   |    |            NP
  |    |    |     _______|______
  N   AUX   V   DET      N      N
  |    |    |    |       |      |
Marge will make  a      ham sandwitch
```

In [18]:

```
s2= nltk.Tree.fromstring('(S (AUX will) (NP (N Marge)) (VP (V make) (NP (DET a) (
s2.pretty_print()
```

```
            S
  __________|________
 |     |             VP
 |     |     ________|___
 |     NP   |            NP
 |     |    |     _______|______
AUX    N    V   DET      N      N
 |     |    |    |       |      |
will Marge make  a      ham sandwitch
```

### Exercise-4

In [21]:
```
s3= nltk.Tree.fromstring('(S (NP Homer) (VP ate (NP (DET the) (N donut)) (PP on (
s3.pretty_print()
```

```
       S
   ____|____________
  |                 VP
  |     ____________|________
  |    |       |             PP
  |    |       |          ___|___
  |    |       NP        |       NP
  |    |    ___|____     |    ___|____
  NP   |  DET       N    |  DET       NP
  |    |   |        |    |   |        |
Homer ate the     donut  on the     table
```

### Exercise-5

In [20]:
```
s4= nltk.Tree.fromstring('(S (NP my old cat) (VP died (PP on (NP tuesday))))')
s4.pretty_print()
```

```
         S
      ___|________
     |            VP
     |        ____|___
     |       |        PP
     |       |     ___|_____
     NP      |    |         NP
  ___|___    |    |         |
 my old cat died  on     tuesday
```

In [22]:
```
s5= nltk.Tree.fromstring('(S (NP (N children)) (AUX must) (VP (VP (V play)) (PP (
s5.pretty_print()
```

```
               S
    ___________|________
   |      |             VP
   |      |     ________|____
   |      |    |             PP
   |      |    |     ________|____
   NP     |    VP   |             NP
   |      |    |    |         ____|_____
   N     AUX   V    P       DET         N
   |      |    |    |        |          |
children must play  in     their     friends
```

### Exercise 6

In [25]:
```
print(ec)
```

```
(VP (V make) (NP (DET a) (N ham) (N sandwich)))
```

In [26]:
```
vp_rules= ec.productions() # list of all CF rules used in the tree
vp_rules
```

Out[26]:
```
[VP -> V NP,
 V -> 'make',
 NP -> DET N N,
 DET -> 'a',
 N -> 'ham',
 N -> 'sandwich']
```

In [27]:
```
vp_rules[0]
```

Out[27]:
```
VP -> V NP
```

In [28]:
```
vp_rules[1]
```

Out[28]:
```
V -> 'make'
```

In [29]:
```
vp_rules[0].is_lexical()
```

Out[29]:
```
False
```

In [30]:
```
vp_rules[1].is_lexical()
```

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

### Explore the CF rules of s5

In [31]:
```
print(s5)
```

```
(S
  (NP (N children))
  (AUX must)
  (VP (VP (V play)) (PP (P in) (NP (DET their) (N friends)))))
```

In [32]:
```
s5_rules= s5.productions()
s5_rules
```

Out[32]:
```
[S -> NP AUX VP,
 NP -> N,
 N -> 'children',
 AUX -> 'must',
 VP -> VP PP,
 VP -> V,
 V -> 'play',
 PP -> P NP,
 P -> 'in',
 NP -> DET N,
 DET -> 'their',
 N -> 'friends']
```

### a. How many CF rules are used in s5?

In [33]:
```
print("How many CF values are used in s5 ",len(s5_rules))
```

```
How many CF values are used in s5  12
```

### b. How many unique CF rules are used in s5?

In [34]:
```
r= npt.array(s5_rules)
print("How many unique CF rules are used in s5 ",len(npt.unique(r)))
```

```
How many unique CF rules are used in s5  12
```

### c. How many of them are lexical?

In [36]:
```
n= 0
for r in s5_rules:
    if r.is_lexical():
        n= n+1
        print("How many of them are lexical? ",n)
```

```
How many of them are lexical?  1
How many of them are lexical?  2
How many of them are lexical?  3
How many of them are lexical?  4
How many of them are lexical?  5
How many of them are lexical?  6
```

In [ ]:
```

```