NAME : RETHINAGIRI G

ROLL NO : 225229130

COURSE TITLE : NATURAL LANGUAGE PRE-PROCESSING LAB

LAB_11. Building Parse Trees1

Exercise-I

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
```

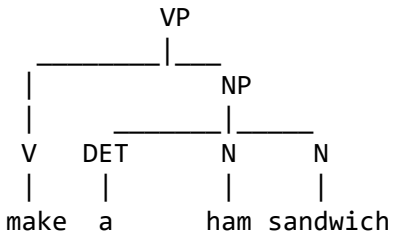
In [2]:
```
np= nltk.Tree.fromstring('(NP (N Marge))')
np.pretty_print()
```

```
  NP
  |
  N
  |
Marge
```

In [3]:
```
ar= nltk.Tree.fromstring('(AUX will)')
ar.pretty_print()
```

```
AUX
 |
will
```

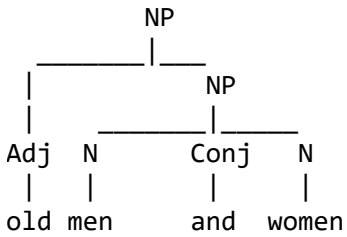
In [4]:
```
tk= nltk.Tree.fromstring('(VP (V make) (NP (DET a) (N ham) (N sandwich)))')
tk.pretty_print()
```



Exercise-II

Create a parse tree for phrase old men and women. Is it _well formed sentence or ambiguous sentence?.

In [5]:
```
text1 = nltk.Tree.fromstring('(NP (Adj old) (NP (N men) (Conj and) (N women)))')
text1.pretty_print()
```

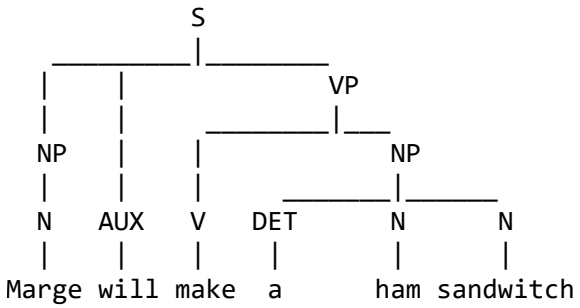


This is a well-formed sentence, as it follows the basic structure of subject-verb-object (though the verb is implied). It is not ambiguous, as the structure makes it clear that both "men" and "women" are being described as "old."

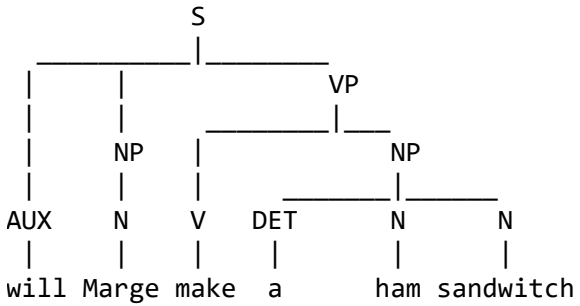
Exercise-III

Using them,build two tree objects,named s1 and s2, for the following sentences.The trees should look exactly like the ones shown below

In [6]:
```
s1= nltk.Tree.fromstring('(S (NP (N Marge)) (AUX will) (VP (V make) (NP (DET a) (N ham) (N sandwitch))))')
s1.pretty_print()
```



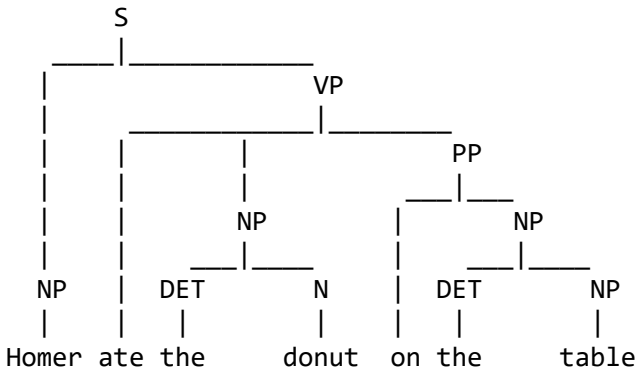
In [7]:
```
s2= nltk.Tree.fromstring('(S (AUX will) (NP (N Marge)) (VP (V make) (NP (DET a) (N ham) (N sandwitch))))')
s2.pretty_print()
```



Exercise-4

Build a tree object named s3 for the following sentence,using its full-sentence string representation.

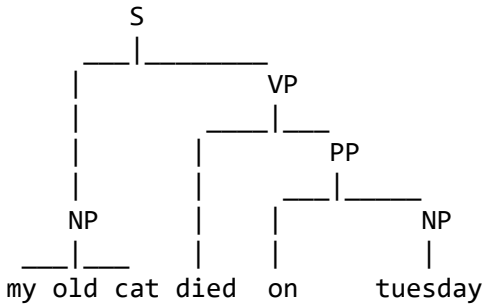
In [8]:
```
s3= nltk.Tree.fromstring('(S (NP Homer) (VP ate (NP (DET the) (N donut)) (PP on (NP (DET the)(NP table)))))')
s3.pretty_print()
```



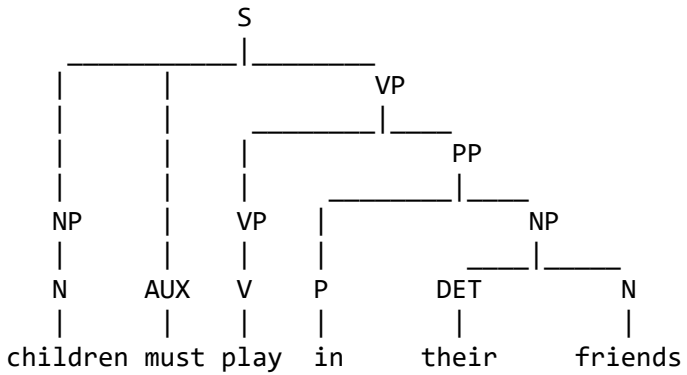
Exercise-5

Build tree objects named s4 and s5 for the following sentences.

In [9]:
```
s4= nltk.Tree.fromstring('(S (NP my old cat) (VP died (PP on (NP tuesday))))')
s4.pretty_print()
```

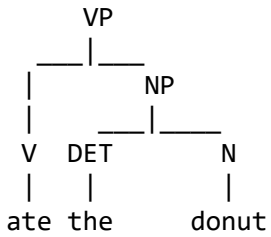


In [10]:
```
s5= nltk.Tree.fromstring('(S (NP (N children)) (AUX must) (VP (VP (V play)) (PP (P in) (NP(DET their) (N friends)))))')
s5.pretty_print()
```



Exercise-6

In [11]:
```
vp= nltk.Tree.fromstring('(VP (V ate) (NP (DET the) (N donut)))')
vp.pretty_print()
```



In [12]:
```
print(vp)
```

```
(VP (V ate) (NP (DET the) (N donut)))
```

In [13]:
```
vp_rules=vp.productions()
```

In [14]:
```
vp_rules
```

Out[14]:
```
[VP -> V NP, V -> 'ate', NP -> DET N, DET -> 'the', N -> 'donut']
```

In [15]:
```
vp_rules[0]
```

Out[15]:
```
VP -> V NP
```

In [16]:
```
vp_rules[1]
```

Out[16]:
```
V -> 'ate'
```

In [17]:
```
vp_rules[0].is_lexical()
```

Out[17]:
```
False
```

In [18]:
```
vp_rules[1].is_lexical()
```

Out[18]:
```
True
```

Explore the CF rules of s5.Include in your script the answers to the following:

In [19]:
```
s5_rules=s5.productions()
s5_rules
```

Out[19]:
```
[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 [20]:
```
len(s5_rules)
```

Out[20]:
```
12
```

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

In [21]:
```
st= npt.array(s5_rules)
len(npt.unique(st))
```

Out[21]: 12

c. How many of them are lexical?

In [22]:
```
n= 0
for i in s5_rules:
    if i.is_lexical():
        n= n+1
        print(n)
```

```
1
2
3
4
5
6
```

In [ ]: