Shri Ramdeobaba College of Engineering and Nagpur

Department of Computer Science and Engineering

Natural Language Processing Lab

Name : Shantanu Mane

Branch : CSE - AIML (VIth SEM)

Roll Num : E-63

Importing the Dependencies

```
import nltk
from nltk import RegexpParser
from nltk.corpus.reader import ChunkedCorpusReader
from nltk.corpus import state_union
```

Chunking

Using the `RegexpParser()`

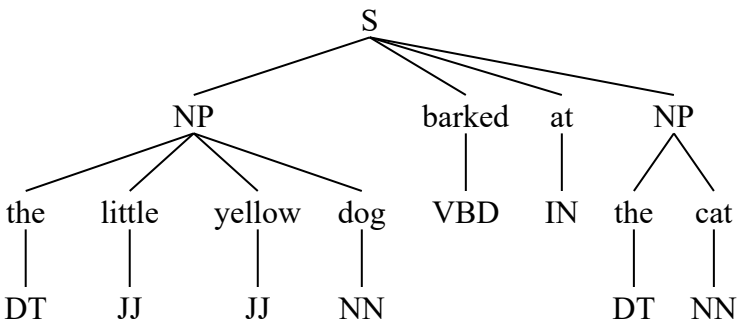
```
sentence = "the little yellow dog barked at the cat"

tokenized_sentence = nltk.word_tokenize(sentence)

tags = nltk.pos_tag(tokenized_sentence)

grammar = "NP: {<DT>?<JJ>*<NN>}"

regParser = RegexpParser(grammar)
result = regParser.parse(tags)

print(result)
```

```
(S
  (NP the/DT little/JJ yellow/JJ dog/NN)
  barked/VBD
  at/IN
  (NP the/DT cat/NN))
```

Drawing the Parse Tree

```
result
```

```
Tree('S', [Tree('NP', [('the', 'DT'), ('little', 'JJ'), ('yellow', 'JJ'), ('dog', 'NN')]), ('barked', 'VBD'), ('at', 'IN'), Tree(
```




```
from nltk.corpus import brown
```

Using the `ChunkedCorpusReader()`

```
chunkCorpusReader = ChunkedCorpusReader('.', r'NP: {<DT>?<JJ>*<NN>}')

chunks = chunkCorpusReader.chunked_words('../data/sentence.txt')

print(f'Chunked Words : {chunks}')
```

```
Chunked Words : [('the', None), ('little', None), ('yellow', None), ...]
```

```
tagged_sentence = chunkCorpusReader.chunked_sents('../data/sentence.txt')
tagged_sentence
```

```
[Tree('S', [('the', None), ('little', None), ('yellow', None), ('dog', None), ('barked', None), ('at', None), ('the', None), ('ca
```

```
grammar = r"""
NP:
{<.*>+}
}<VBD|IN>+{
"""

regParser = RegexpParser(grammar)
regParser.parse(tags)
```

```
Tree('S', [Tree('NP', [('the', 'DT'), ('little', 'JJ'), ('yellow', 'JJ'), ('dog', 'NN')]), ('barked', 'VBD'), ('at', 'IN'), Tree(
```

Part A

```
chunk_grammer = r"""
NP: {<DT>?<JJ>*<NN>}
"""
```

```
chunk_parser = RegexpParser(chunk_grammer)
```

```
sentences = nltk.sent_tokenize(state_union.raw())
sentence = sentences[3]
tokens = nltk.word_tokenize(sentence)
tagged = nltk.pos_tag(tokens)
```

```
sentence
```

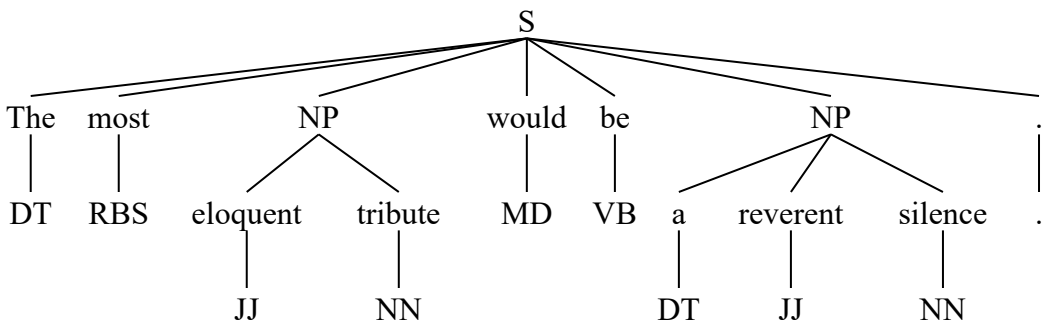
```
'The most eloquent tribute would be a reverent silence.'
```

```
tokens
```

```
['The',
 'most',
 'eloquent',
 'tribute',
 'would',
 'be',
 'a',
 'reverent',
 'silence',
 '.']
```

```
tagged
```

```
                              [('The', 'DT'),
 ('most', 'RBS'),
 ('eloquent', 'JJ'),
 ('tribute', 'NN'),
 ('would', 'MD'),
 ('be', 'VB'),
 ('a', 'DT'),
 ('reverent', 'JJ'),
 ('silence', 'NN'),
 ('.', '.')]
```




```
chunk_parser.parse(tagged)
```

```
Tree('S', [('The', 'DT'), ('most', 'RBS'), Tree('NP', [('eloquent', 'JJ'), ('tribute', 'NN')]), ('would', 'MD'), ('be', 'VB'), Tr
```

Part B

```
chink_grammer = r"""
 NP:
 {<.*>+}
 }<VB.*>{
"""
```

```
chink_parser = RegexpParser(chink_grammer)
```

```
chink_parser.parse(tagged)
```

```
Tree('S', [Tree('NP', [('The', 'DT'), ('most', 'RBS'), ('eloquent', 'JJ'), ('tribute', 'NN'), ('would', 'MD')]), ('be', 'VB'), Tr
```