Challenge 3: Parsing

1

Implement a simple top-down parser along the lines of the Parse procedure given in class. Add control printouts whenever Parse is called and whenever the procedure returns.

2

Implement a bottom-up parser which realizes the simple bottom-up parsing schema given in class. Again, add printouts whenever an item is created. Provide a documented listing of the algorithm.

3

Run the two parsers on the following grammar:

```
S → NP VP              Det → that | this | a | the
S → Aux NP VP          Noun → book | flight | meal
S → VP                 Verb → book | include | includes
VP → Verb              Prep → from | to | on
VP → Verb NP           Proper-Noun → JFK | LA | TWA
NP → Det Nominal       Aux → does
NP → Proper-Noun       Nominal → Noun
Nominal → Noun Nominal
Nominal → Nominal PP
PP → Prep NP
```

Try to parse the following strings:

- this flight includes a meal
- the flight from LA includes a meal
- does the flight include a meal
- book that flight
the flight from JFK to LA on TWA includes a meal
the flight does include a meal
the flight does not include a meal
the flight

Based on these experiments, compare the two parsers in terms of completeness, efficiency, reduplication of effort etc.

4

Propose simple modifications to each of the parsers which would guarantee termination. You do not have to implement anything here.

5

Propose an extension to each of the parsers which would result in printing one tree induced by the grammar on the input sentence. There is no need to account for all possible trees. You do not have to implement anything here.