(1) Implement a probabilistic parser using the CYK algorithm. Use the following grammar and input sentences to test your parser:

S → NP VP	[1.0]	DET → the	[0.8]
NP → DET N	[0.8]	DET → a	[0.2]
NP → NP PP	[0.2]	N → student	[0.55]
VP → V NP	[0.4]	N → book	[0.25]
VP → VP PP	[0.6]	N → library	[0.2]
PP → P NP	[1.0]	V → reads	[1.0]

On the course homepage, you can find a partial implementation (in Python) which you might find useful.

(2) On the course web-page you can download a relative large grammar where rules are annotated with frequency counts. Convert this grammar into a PCFG in Chomsky normal form and apply your parser implementation from (1) to the resulting grammar. Use the following sentences as test inputs, and compute labelled precision and labelled recall for the parser output.

a. The reason was not high interest rates or labor costs

b. Many other factors played a part in yesterday 's comeback

The corresponding gold-standard trees are as follows:

c. [S [NP [DT The] [NN reason]]
   [VP [VBD was]
      [RB not]
      [NP [JJ high] [NN interest] [NNS rates] [CC or] [NN labor] [NNS costs]]]]

d. [S [NP [JJ Many] [JJ other] [NNS factors]]
   [VP [VBD played]
      [NP [DT a] [NN part]]
      [PP [IN in] [NP [NP [NN yesterday] [POS 's]] [NN comeback]]]]]

Note 1: you don't have to implement anything to compute labelled precision and labelled recall – you can do the computation manually using pen & paper.

Note 2: the gold standard trees are not binary, so you have to "undo" the binarisation of the parser output in a first step.