1.5 Parsing

1. Soit la grammaire $S \to aB \mid bS \mid cC \mid c \mid bD$
   $B \to aS$
   $C \to aC \mid a$
   $D \to aB \mid bC \mid b$

   Détailler l'analyse descendante que l'on peut faire de *bbb*. Proposer une grammaire reconnaissant le même langage qui analyse le mot *bbb* sans retour-arrière (*backtrack*).
2. Soit la grammaire hors contexte suivante :

S	$\to$	p
p	$\to$	gn v1 *que* p | gn v2
gn	$\to$	np | det nc
np	$\to$	*Léa* | *Luc* | *Ève* | *Max*
nc	$\to$	*femme* | *homme* | *étudiante* | *étudiant* | *fille* | *garçon*
det	$\to$	*le* | *la* | *l'*
v1	$\to$	*pense* | *croit* | *voit* | *sait* | *dit* | *raconte*
v2	$\to$	*se promène* | *marche* | *part*

   (a) Donner quatre phrases distinctes reconnues par cette grammaire, contenant respectivement 0, 1, 2 et 3 fois le mot *que*.
   (b) Pour quelles raisons ces phrases ne sont-elles pas toutes correctes en français ? Comment modifier la grammaire pour corriger cela ?
   (c) Donner l'arbre de dérivation de *Luc sait que la femme croit que Léa part.*
   (d) Donner les branches pertinentes de l'arbre d'exploration des solutions (comme vu en cours) pour une analyse descendante de la suite *Luc pense que Léa se promène.*
3. Soit la grammaire $S \to aSb$; $S \to cC$; $C \to dC$; $C \to c$. Ébaucher l'analyse descendante pour le mot *acddcb*. Peut-on trouver un moyen simple de prédire de façon systématique la bonne dérivation ?

Grammaires prédictives LL(k)

grammaires SLL(1) Grammaires dont toutes les productions sont de la forme $A \to a\alpha$ (avec $a \in X$, et $\alpha \in (X \cup V)^*$) ; et telles que s'il existe deux dérivations $A \to a_1\alpha_1$ et $A \to a_2\alpha_2$, alors $a_1 \neq a_2$.
Dans ce cas, il suffit de regarder un symbole (*lookahead-1*) en avant pour décider quelle dérivation s'applique. L'analyse descendante devient linéaire. Problème : toutes les grammaires algébriques n'ont pas une grammaire SSL(1) équivalente.

grammaires en forme normale de Greibach Grammaires dont toutes les productions sont de la forme $A \to a\alpha$ (avec $a \in X$, et $\alpha \in (X \cup V)^*$).
Cette fois, toute grammaire algébrique peut être « normalisée ». L'analyse est polynomiale, mais l'arbre syntaxique est très différent de celui de la grammaire initiale.

grammaires LL(k) L'idée est de construire, à partir de la grammaire initiale (sans la modifier), une **table de prédiction** : étant donné le non-terminal à réduire, et le symbole courant du mot à reconnaître, cette table donne toutes les dérivations possibles.
S'il n'y a qu'une dérivation possible dans chaque cas (au plus), la grammaire est dite LL(1). Si on peut construire une table où il n'y a qu'une dérivation possible en regardant 2 caractères, la grammaire est dite LL(2).
Les grammaires LL(k) induisent une hiérarchie stricte de langages.