Compilation - N3, S5
TP 2 : Génération de code avec ANTLR

1. L'objectif de ce TP c'est la génération de code avec ANTLR
2. Votre ami pour la documentation de ANTLR :
   `https://github.com/antlr/antlr4/blob/master/doc/index.md`
3. Se référer à la fiche de TP 1 pour l'installation et l'utilisation de ANTLR

Question 2.1 Écrire une grammaire `Expr_Bool` qui permet de générer des expressions booléennes. On acceptera les opérateurs suivants : *and*, *or* et *not*, ainsi que des expressions parenthésées. Rappelons les priorités *not* > *and* > *or*. Vous pourrez tester avec les expressions suivantes :

```
true and false or true and not true
true or false and false
true or (false or false)
false or (true and not false)
(not true and false) or (true and not false)
```

Question 2.2 Modifier la grammaire `Expr_Bool` pour qu'un programme puisse être une suite d'expressions qui sont séparées par des sauts de ligne et/ou des `;`.

Question 2.3 Un enseignant propose la grammaire `G` suivante et affirme qu'elle génère bien les expressions booléennes.

```
S -> true | false | not S | and S S | or S S
```

Attention : l'enseignant n'affirme pas que pour toute expression booléenne, il existe un unique mot dans `L(G)` qui lui est équivalent. Il affirme juste que pour chaque expression booléenne, il existe un mot dans `L(G)` ayant la même valeur de vérité.

Décorez la grammaire `Expr_Bool` pour qu'elle génère, pour chaque expression booléenne, un mot équivalent dans `L(G)`. Si un programme est constitué de plusieurs expressions booléennes, les mots associés sont séparés par des sauts de ligne.

Question 2.4* L'objectif ici c'est d'écrire un simulateur d'une machine qui permet d'évaluer des expressions booléennes.

1. Écrire une grammaire attribuée `Eval_G`, qui est une décoration de la grammaire `G` de la question précédente et qui calcule pour chaque mot sa valeur booléenne. Veillez à ce qu'un programme soit une suite de mots séparés par des sauts de ligne.
2. Modifiez la grammaire attribuée `Eval_G` pour que la valeur de chaque mot soit affichée ainsi : `valeur_expr_N=v` où `N` est le numéro de ligne du mot et `v` sa valeur. Une valeur affichée par ligne.
3. Vous pouvez maintenant utiliser le compilateur de `Eval_G` comme simulateur d'une machine exécutant les programmes écrits dans le langage de `Expr_Bool`.