Compiladores – Lista #4

1. Construa um analisador sintático recursivo-descendente com retrocesso para a gramática

$$S \to aSbS \mid bSaS \mid \epsilon$$

   É possível construir um analisador sintático preditivo para esta gramática?

2. Construa um analisador sintático preditivo para a gramática

$$\begin{aligned} bexpr &\to bexpr \textbf{ or } btermo \mid btermo \\ btermo &\to btermo \textbf{ and } bfator \mid bfator \\ bfator &\to \textbf{not } bfator \mid (\ bexpr\ ) \mid \textbf{true} \mid \textbf{false} \end{aligned}$$

3. Para a expressão de entrada `(4 × 7 + 1) × 2`, construa uma árvore gramatical anotada de acordo com a definição dirigida pela sintaxe:

Produção	Regra semântica
$L \to E$ **n**	imprimir($E.val$)
$E \to E_1$ + T	$E.val := E_1.val + T.val$
$E \to T$	$E.val := T.val$
$T \to T_1$ × F	$T.val := T_1.val \times F.val$
$T \to F$	$T.val := F.val$
$F \to (E)$	$F.val := E.val$
$F \to$ **digito**	$F.val :=$ **digito**$.lexval$

4. Crie o DAG para a expressão `a + a + (a + a + a + (a + a + a + a))`, assumindo que o operador + seja associativo à esquerda.

5. A gramática abaixo gera expressões formadas pela aplicação do operador aritmético + a constantes inteiras e reais. Quando dois inteiros são adicionados, o tipo resultante é inteiro, caso contrário é real.

$$\begin{aligned} E &\to E + T \mid T \\ T &\to \textbf{num.num} \mid \textbf{num} \end{aligned}$$

   Dê uma tradução dirigida pela sintaxe para determinar o tipo de cada subexpressão.