Departamento de Matemática da Universidade de Coimbra		
2004/2005	**Compiladores**	**Folha 3** (v1.4)

1. Construa especificações *Bison* capazes de:
   (a) calcular o sumatório de uma lista de inteiros (separados por ','');
   (b) calcular o valor de uma expressão aritmética simples (uma só operação).
   (c) calcular o valor de uma expressão aritmética em notação pós-fixa, com as operações elementares $+, -, *, /$.

2. Pretende-se construir um programa que reconheça e calcule o valor de expressões aritméticas elementares:
   (a) Escreva uma sintaxe para expressões algébricas com as quatro operações básicas: $+, -, *, /$, parêntesis, a expressão simétrica e números inteiros (sem se preocupar com precedências).
   (b) Escreva uma árvore de derivação para a expressão: $2 + 3 * 4$. Esta árvore de derivação é única? Que tipo de conflito está aqui em causa? Explique.
   (c) Altere a gramática de forma a que deixe de ser ambígua, a) factorizando-a; b) utilizando precedências para as operações.
   (d) Adicione acções a cada regra que permita calcular o resultado total de uma expressão com base nos valores dos nós da árvore sintáctica.
   (e) Altere as acções do programa anterior para, em vez de calcular o valor da expressão, construa uma árvore sintáctica que represente a mesma. Descreva as estruturas de dados necessárias.
   (f) Adicione a cada estrutura que compõe a árvore sintáctica uma função que permita calcular o valor do nó correspondente com base nos valores que contém.
   (g) Escreva uma função que, dada uma árvore sintáctica correspondente a uma expressão, a desça recursivamente e calcule o seu valor. Deve utilizar para o efeito as funções definidas na alínea anterior.

3. Projetar uma gramática não ambígua para uma linguagem de expressões lógicas usando as constantes «Verdade» e «Falso». A linguagem deverá conter os operadores booleanos de negação (não), conjunção (e) e disjunção (ou), com as seguintes características (dados em ordem de menor a maior precedência):

Operador	Características
e	Binário, infixo
ou	Binário, infixo
não	Unário, posfixo
( )	Unário

4. Projetar uma gramática não ambígua para a linguagem das expressões regulares que poden ser construídas usando o alfabeto 0, 1. Os operadores que devem poder ser usados nas expressões regulares definidas por esta gramática são os seguintes (dados em ordem de menor a maior precedência):

Sintaxe	Operação	Número de operandos	Associatividade
R|S	União	binária	Assoc. a esquerda
RS	Concatenação	binária	Assoc. a direita
R+ , R*	Fechos	unárias	
(R)	Parênteses	unária	

Alguns exemplos de expressões regulares que podem ser obtidas usando esta gramática são:

```
010     (01*|(0|1+)*|1)101     1(0(0+)1|01(0+)*1)*1111     (11000*+*)+**
```

5. Escrever tabelas de analizadores sintáticos SLR para as linguagens definidas pelas seguintes gramáticas: (aonde letras maiúsculas representam símbolos não terminais, letras minúsculas representam símbolos terminais):

$S \rightarrow \text{id}(L)$
$S \rightarrow \text{id}$
$L \rightarrow \varepsilon$
$L \rightarrow S\ Q$
$Q \rightarrow \varepsilon$
$Q \rightarrow \text{, } S\ Q$

$S \rightarrow S$ inst
$S \rightarrow T\ R\ V$
$T \rightarrow$ tipo
$T \rightarrow \varepsilon$
$R \rightarrow$ blq V fblq
$R \rightarrow \varepsilon$
$V \rightarrow$ id S fin
$V \rightarrow$ id ;
$V \rightarrow \varepsilon$

$E \rightarrow [L]$
$E \rightarrow$ a
$L \rightarrow E\ Q$
$Q \rightarrow \text{, } L$
$Q \rightarrow \varepsilon$
$V \rightarrow \varepsilon$

$S \rightarrow S$ inst
$S \rightarrow S$ var D
$S \rightarrow \varepsilon$
$D \rightarrow D$ ident E
$D \rightarrow D$ ident sep
$D \rightarrow$ int
$D \rightarrow$ float
$E \rightarrow S$ fproc

$S \rightarrow U$
$S \rightarrow M$
$M \rightarrow$ i M e M
$M \rightarrow$ a
$U \rightarrow$ i S
$U \rightarrow M$ e U

$P \rightarrow P\ C$
$P \rightarrow [C]$
$P \rightarrow \varepsilon$
$C \rightarrow \{P\}$
$C \rightarrow (P)$

6. Implementar um analisador sintáctico para uma variante da linguagem C, sem directivas de pre-processador, estruturas de dados e apontadores. Os únicos tipos base são o `bool`, `int`, `float`, `string`. Comece pelos construtores básicos: `for`, `while`, `break`, `if`, `then`, `else`, ... e adicione a atribuição e expressões numéricas com inteiros e variáveis.