Centro Universitário de Belo Horizonte – Uni-BH
Departamento de Ciências Exatas e Tecnologia
Curso: Ciência da Computação
Disciplina: Compiladores
Professor: Moisés Henrique Ramos Pereira

Lista de Exercícios I para a Prova AIA – 19/04/2013 – Valor: 10 pontos

1) Explique, sucintamente, os componentes de um compilador e suas relações. Qual a diferença entre compilador e interpretador?

2) Explique o que é *token*, *padrão de formatação* e *lexema*. Dê um exemplo de token na linguagem C/C++ (exceto palavra reservada), seu respectivo padrão e um exemplo de lexema deste token.

3) Qual é o papel da Tabela de Símbolos na Análise Léxica?

4) Para o trecho de programa fonte a seguir e considerando os padrões do trabalho prático, indique a sequencia de lexemas e tokens que um analisador léxico identifica:

a) total = soma * 2;

b)
```
public class Principal {
        public static void main (String[] args) {
                System.out.println("Hello, world!"); }}
```

5) Em uma linguagem de programação, comentários de mais de uma linha são definidos por /* seguido por caracteres (qualquer caractere) e finalizado por */. Mostre um diagrama de transição (AFD) que reconheça comentários nesta linguagem.

6) Nas gramáticas a seguir, realize as modificações necessárias para que seja possível implementar um parser LL(1) para cada uma.

a) stmt → **if** expr **then** stmtList **endif**;
stmt → **if** expr **then** stmtList else stmtList **endif**;

b) E → E + E | E * E | a

7) Considere a gramática a seguir (já com as regras enumeradas):

program → **begin** statementList **end** [1]
statementList → statement statementTail [2]
statementTail → statement statementTail [3] | λ [4]
statement → id := expression; [5] | **read** (idList); [6] | **write** (exprList); [7]
idList → id idTtail [8]
idTail → , id idTail [9] | λ [10]
exprList → expression exprTail [11]
exprTail → , expression exprTail [12] | λ [13]
expression → primary primaryTail [14]
primaryTail → addOp primary primaryTail [15] | λ [16]
primary → (expression) [17] | id [18] | intLiteral [19]
addOp → + [20] | - [21]

Na gramática, o símbolo *id* denota o padrão de formação de identificadores, e o símbolo *intLiteral* denota o padrão de formação de constantes numéricas inteiras.

a) Quais são os tokens da linguagem?

b) Quais dos programas abaixo estão sintaticamente corretos nesta linguagem e quais não estão. Aponte os erros sintáticos encontrados.

1) begin
```
    read(x, t);
    a := b + 5 – (x – 7 + t);
    write(a);
end
```

2) begin
```
    a := -b + 5 – x – 7 + t;
    write(a + 2);
    x := x – 3 v
end;
```

c) Compute os conjuntos FIRST e FOLLOW para os símbolos não-terminais.

d) Mostre uma implementação do procedimento do parser preditivo recursivo para o simbolo *statement.*

8) Construa um parser descendente, predizível e recursivo para a gramática :

S → (L) | a
L → L , S | S

9) Considere a gramática:

S → STL; | λ
T → *int* | *float*
L → L, *id* | L, *id*[*num*] | *id* | *id*[*num*]

a) Elimine recursividade à esquerda e fatore a gramática .

b) Determine os conjuntos FOLLOW e FIRST para a gramática obtida no item 2b.

c) Construa uma tabela de *parser* para a gramática obtida em 2b.

d) Mostre a execução do algoritmo preditivo não-recursivo para reconhecer *int casa[10], i;*

10) Qual é o papel da Tabela de Símbolos na Análise Sintática?

11) Execute o algoritmo parser ascendente *Shift-Reduce* para a gramática abaixo utilizando-se a entrada ((a , a)).

S' → S
S → (L) | a
L → L , S | S