UNIP – Universidade Paulista	
Curso.....:	Bach. em Ciência da Computação
Disciplina..:	Compiladores e Computabilidade
Professor...:	Leandro Carlos Fernandes

-:: Lista de Exercícios #1 ::-

Tópicos: Visão Geral, Análise Léxica e Introdução à An. Sintática

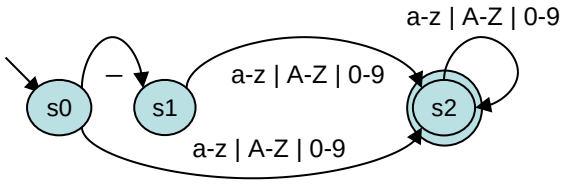
1) Uma das primeiras atividades que um programador realiza é a preparação de seu computador, criando um ambiente de desenvolvimento. Isto é, fazendo a instalação de várias ferramentas que o ajudarão na tarefa de escrever, compilar e executar os programas que irá criar.
   Quando instalamos aplicativos de programação tais como o Eclipse, NetBeans, J2SE, o Framework .NET, o Visual Studio, o que dentre este conjunto realmente é o compilador e o que são as ferramentas de desenvolvimento?

2) O processo que envolve a tradução do código-fonte em seu correspondente executável envolve uma série de tarefas que precisam ser realizadas pelo software tradutor. Dada a complexidade deste processo, essas tarefas são divididas em etapas, tendo certos objetivos e encadeadas de modo a produzirem um código-alvo equivalente ao fonte.
   Atualmente a construção de compiladores segue o chamado modelo de Análise e Síntese. Enumere cada fase destas duas grandes etapas e explique quais tarefas são executadas em cada uma delas.

3) Embora muito as utilizem como sinônimos, é sabido que existem diferenças entre os termos *Compilador*, *Montador* e *Tradutor*. Qual(is) seria(m) ela(s)?

4) A tarefa de identificar os lexemas da linguagem (tokens) desempenhada pelo Analisador Léxico é orientada por um AFD (autômato finito determinístico). A esse respeito:
   a) Defina um autômato capaz de reconhecer qualquer seqüência numérica de símbolos, quer ela represente um número inteiro ou real, positivo ou negativo.

   b) Considerando o autômato de reconhecimento dado ao lado, processe o trecho de código dado a seguir, comentando detalhadamente seu funcionamento.
      Código-fonte: `_Exemplo 4;`



5) Supondo a descrição a seguir sobre a linguagem LPD (Linguagem de Programação Didática) e que seus elementos são classificados da seguinte maneira:
   - Palavras reservadas: `program, var, int, char, void, return, begin, end, if, then, else, for, to, do, repeat, until, and, or, not`
   - Delimitadores: `{ } comentários, “ ” strings, ‘ ‘ caracteres`
   - Operadores: `* + – / <- = != > >= < <=`
   - Outros símbolos: `. , ; ( ) [ ]`

   Faça a Análise Léxica do programa abaixo, construindo a *lista de tokens* resultante e reportando eventuais erros durante o processo.

```
{este programa corresponde
a umas das questões da lista
program exer_list;
}var int i[];
begin if (for())
then x.3 else
y<--0,9 end.
```

6) A estrutura gramatical do código-fonte é analisada pelo Analisador Sintático (ou *Parser*), que tem por objetivo principal verificar se os comandos atendem a sintaxe definida pela linguagem. Esta tarefa é feita tentando-se construir a árvore de derivação (árvore sintática) do programa.
   Suponha uma linguagem de programação hipotética, com sintaxe parecida com C (C-*liked*) e considere que a gramática a seguir especifica a estrutura de seu comando FOR.

```
G=(Vn,Vt,P,<cmd>)
P: <cmd>::= for(<decvar>;<cond>;<cont>) |
           for(<decvar>;<cond>;)  |
           for(;<cond>;<cont>)  |
           for(;<cond>;)
   <decvar>::= int <var> = <num>
   <cond>::= <var> > <termo> | <var> < <termo> | <var> != <termo> |
             <termo> > <var> | <termo> < <var> | <termo> != <var>
   <cont>::= <var>++  |  ++<var>
   <termo>::= <var> | <num>
   <var>::= i | j | k
   <num>::= <dig>{<dig>}
   <dig>::= 0|1|2|3|4|5|6|7|8|9
```

Dê a árvore de derivação para cada uma das cadeias a seguir:

a) `for(int i=1;i<10;i++)`
b) `for(;i<10;i++)`
c) `for(;i<10;)`

7) Uma característica indesejável a qualquer linguagem de programação é a possiblidade de interpretação dúbia de alguma de suas instruções, o que causaria um funcionamento/comportamento do programa diferente do que foi programado pelo desenvolvedor. Analise a gramática abaixo e responda:

```
G=(Vn,Vt,P,<cmd>)
Vn = {<cmd>, <cond>}
Vt = {if, then, else, ;, cond1, cond2, cond3, comand1, comand2, comand3}
P: <cmd>::= if<cond>then<cmd>; | if<cond>then<cmd>else<cmd> |
            comand1 | comand2 | comand3
   <cond>::= cond1 | cond2 | cond3
```

a) Dê a arvore de derivação para a cadeia:

```
if cond1
  then comand1
  else if cond2
      then if cond3
            then comand2;
            else comand3
```

b) Trata-se de uma gramática ambígua? Em caso positivo, como poderia ser resolvido o problema?