COMPILADORES

Prova 1 – 14/11/2019 – Prof. Marcus Ramos

Questão 1 (1 ponto): A linguagem-fonte de um compilador pode ser de baixo-nível? E a linguagem-objeto, pode ser de alto-nível? Esclareça os conceitos envolvidos nestas perguntas.

Questão 2 (1 ponto): Usando a notação dos diagramas-T, mostre como acontece a compilação e a execução de um programa escrito na linguagem Java numa máquina x86.

Questão 3 (1 ponto): Justifique o interesse pelo uso da técnica de bootstrapping na construção de compiladores.

Questão 4 (1 ponto): Qual a principal vantagem de se organizar um compilador em vários passos ao invés de um único passo?

Questão 5 (1 ponto): Quais as conseqüências práticas de se representar a sintaxe de uma linguagem de programação por meio de uma gramática livre de contexto?

Questão 6 (1 ponto): Prove que a seguinte gramática é LL(1):

$S \rightarrow abX \mid Ycba \mid bXa \mid Xba$

$X \rightarrow dX \mid e$

$Y \rightarrow fX \mid gX \mid hX \mid \varepsilon$

Questão 7 (1 ponto): Descreva, em linhas gerais, como funcionam os métodos de análise sintática descendente e de análise sintática ascendente.

Questão 8 (1 ponto): Que outras tarefas um analisador sintático típico deve realizar além de apenas verificar se o programa-fonte pertence ou não pertence à linguagem-fonte?

Questão 9 (2 pontos): Obtenha um esboço de reconhecedor sintático para a linguagem cuja sintaxe é representada pela gramática abaixo. Considere a existência de um analisador léxico adequado e resolva pendências gramaticais e de especificação da linguagem-fonte.

```
<atribuição> ::= <id> := <expressão>
<comando> ::= <atribuição>
      | <condicional>
      | <iterativo>
      | <comando-composto>
<comando-composto> ::=  begin <lista-de-comandos> end
<condicional> ::= if <expressão> then <comando> ( else <comando>
| <vazio> )
<corpo> ::= < declaração-de-variável > <comando-composto>
<declaração-de-variável> ::= var <lista-de-ids> : <id>
<expressão> ::= <expressão-simples>
      | <expressão-simples> <op-rel> <expressão-simples>
```

```
<expressão-simples> ::= <expressão-simples> <op-ad> <termo>
      | <termo>
<fator> ::= <id>
      | <literal>
      | "(" <expressão> ")"
<iterativo> ::= while <expressão> do <comando>
<lista-de-comandos> ::= <comando> ;
      | <lista-de-comandos> <comando> ;
      | <vazio>
<lista-de-ids> ::= <id>
      | <lista-de-ids> , <id>
<literal> ::= <bool-lit>
      | <int-lit>
      | <float-lit>
<programa> ::= program <id> ; <corpo> .
<termo> ::= <termo> <op-mul> <fator>
      | <fator>
```