Primeira Prova de Compiladores - 2/2002 - 16/12/2002
==========

Indique as afirmações verdadeiras (V) e falsas (F).

1) Uma importante característica de um *assembler* é que uma instrução na linguagem de montagem tem relação direta com uma instrução na linguagem de máquina.
2) Um *scanner* é usado para reconhecer as seqüências de caracteres de um programa e organizá-las e classificá-las em vários tipos de *tokens*.
3) Na análise sintática é verificado se a estrutura das frases do programa está correta.
4) Na análise sintática é verificado, por exemplo, se o número de argumentos passados para um método ou função está correto.
5) A verificação de tipos de um programa é realizada através da análise sintática se a linguagem possui tipos estáticos.
6) Tanto a sintaxe quanto a semântica de um programa podem ser especificadas formalmente.
7) As gramáticas livres de contexto são usadas para especificar a semântica de uma linguagem.
8) A fase do compilador responsável por verificar se o “else” aparece sempre depois de um “if” é a análise contextual.
9) BNF ou EBNF é a notação utilizada para especificar a análise contextual de uma linguagem.
10) A verificação de que uma variável tem que estar definida antes de ser utilizada é feita pela análise contextual.
11) Existem regras automáticas para transformar uma gramática recursiva à esquerda em uma outra que não é recursiva à esquerda.
12) Um compilador pode ser dividido em várias fases.
13) Um compilador pode ser dividido em vários passos.
14) Existem compiladores que possuem apenas uma fase.
15) Árvores Sintáticas Abstratas são utilizadas pela análise contextual para a verificação de tipos.
16) Árvores de parsing (*Parse trees*) servem para entender a análise sintática, mas na prática não são construídas pelo compilador.
17) Árvores de parsing (*Parse trees*) servem para reconhecer as seqüências de letras que formam os *tokens*.
18) Árvores de parsing (*Parse trees*) são a mesma coisa que árvores sintáticas abstratas.
19) Os *parsers top-down* são facilmente desenvolvidos à partir uma gramática recursivas à esquerda.
20) *Recursive descent parsers* são um tipo de *parser top-down* que podem ser facilmente escritos utilizando funções recursivas.
21) Em uma linguagem estaticamente tipada, os erros de tipo são verificados antes de o programa começar a executar.

Dada a especificação abaixo:

```
Program ::= (Statement ";")+

Statement ::= "if" Expression "then" Statement "else" Statement
        | identifier "=" Expression

Expression ::= identifier
         | number
         | identifier "+" Expression
         | number "+" Expression
```

Podemos afirmar que:

22) ela define a sintaxe e semântica contextual de uma mini-linguagem usando uma gramática livre de contexto.
23) A linguagem possui uma produção com recursão à esquerda.
24) A gramática é ambígua.
25) Ela reconhece um programa vazio (em branco).
26) Ela reconhece o programa “if x + 3 then if 3 + 4 then x = 10 + 22 else x = 12;”.

No programa abaixo:

```
let
 const m ~ 2;
 var n : Integer;
 func f (i : Integer) : Integer ~
        i * m
in begin
  n := 10;
  n := let m ~ 3 in f (n);
  print m; print n;
end
```

Podemos afirmar que, ao final da execução do programa:

27) se o escopo da linguagem for estático, o valor impresso de n será 30.
28) se o escopo da linguagem for dinâmico o valor impresso de n será 20.
29) se o escopo da linguagem for dinâmico ao final do programa o valor impresso para m será 3.

30) O processo de bootstrapping é caracterizado pela compilação de uma linguagem utilizando um compilador escrito na própria linguagem.
31) O processo de bootstrapping completo sempre precisa de um compilador inicial escrito em outra linguagem, e que depois deve ser descartado.
32) A diferença entre uma máquina real e uma máquina abastrata é que a máquina abstrata usa um interpretador para executar as instruções de um programa, enquanto que a máquina real possui um processador (CPU) para executar as instruções de um programa.
33) Um *cross-compiler* é um compilador que roda em uma máquina mas gera código para uma outra máquina.
34) Java é uma linguagem que normalmente é compilada para uma máquina abstrata.
35) Entre as vantagens de uso de uma máquina abstrata está o fato de que ela normalmente permite um código mais eficiente que através da compilação para uma máquina real.
36) Sempre existe um processador (CPU) para uma máquina abstrata.
37) O processo de bootstrapping não pode ser usado para melhorar a eficiência de um compilador.
38) O uso de expressões é uma das características de uma linguagem de alto nível.
39) Interpretadores não são podem ser considerados processadores de linguagens, pois não geram código objeto.
40) Não é possível combinar o uso de compiladores e interpretadores na implementação de uma linguagem.

41) (questão que vale dois pontos) Descreva (desenhe) no espaço abaixo, usando os diagramas *tombstone* o processo de compilação e execução de um programa chamado “teste” na linguagem Java usando o JDK.