
	Segunda Avaliação	Nota: 4,5 + 0,5 = 5,0
Curso:	Ciência da Computação	
Disciplina:	Compiladores	
Aluno(a):	Lucas Farias Pereira	Data:

1) Sobre as gramáticas BNF e EBNF marque a afirmativa verdadeira. (1 pt)

- a) () Uma BNF é uma Gramática regular a EBNF é uma GLC
- b) () A BNF é uma GLC enquanto que uma EBNF é uma GR para facilitar a implementação de um analisador sintático.
- c) () A BNF é uma gramática que apresenta recursão a esquerda e não é fatorada a esquerda, já a EBNF não apresenta estas características.
- d) () A BNF é uma forma simplificada da EBNF e que por isso é adequada para implementação de analisadores preditivos.
- e) () A EBNF é uma adaptação da BNF para esta possa ser implementada com analisadores sintáticos com retrocesso (*backtracking*).
- ✶ f) ☒ Nenhuma das anteriores. 

2) Dada a seguinte gramática, que especifica a sintaxe de Tiny, pergunta-se: Qual das opções corresponde a um programa válido em Tiny? Marque-a com um "x". (0,5 pt)

```

programa → decl-sequência
decl-sequência → decl-sequência ; declaração | declaração
declaração → cond-decl | repet-decl | atrib-decl | leit-decl | escr-decl
cond-decl → if exp then decl-sequência end
           | if exp then decl-sequência else decl-sequência end
repet-decl → repeat decl-sequência until exp
atrib-decl → identificador := exp
leit-decl → read identificador
escr-decl → write exp
exp → exp-simples comp-op exp-simples | exp-simples
comp-op → < | =
exp-simples → exp-simples soma termo | termo
soma → + | -
termo → termo mult fator | fator
mult → * | /
fator → (exp) | número | identificador

```

Lucas Farias Pereira

- a) ☐ $n < 10$
- b) ☒ $\text{varx} := 10$ *e*
- c) ☐ $\text{media} > 10$
- d) ☐ $\text{write}(\text{nota})$
- e) ☐ Nenhuma das anteriores

3) Com relação a função *match* do analisador sintático de Tiny marque a afirmativa verdadeira. (1 pt)

- a) ☐ Retorna um ponteiro para o nó raiz da árvore sintática.
- b) ☐ Verifica para cada *string* retornada pelo analisador léxico, se corresponde a um lexema válido de algum tipo de marca da linguagem e emite uma mensagem de erro caso não seja.
- c) ☐ Verifica para cada *string* retornada pelo analisador léxico, se corresponde a um lexema válido de algum tipo de marca da linguagem, mas não emite qualquer mensagem de erro caso não seja.
- d) ☐ Verifica se o *token* retornado pela função *parse()* coincide com o *token* esperado em dado momento pelo analisador sintático.
- e) ☒ Emite uma mensagem de erro caso o *token* retornado pelo analisador léxico não coincida com o *token* esperado. *e*

4) A sintaxe de uma linguagem de programação pode ser definida por meio de uma GLC $G = (V, T, P, S)$. Qual opção apresenta a definição das regras de produção (P) deste tipo de gramática? (0,5 pt)

- × a) ☒ $V \rightarrow (V \cup T)^*$ *e*
- b) ☐ $V^* \rightarrow (V \cup T)^*$
- c) ☐ $(V \cup T) \rightarrow (V \cup T)^*$
- d) ☐ $V \rightarrow V^*$
- e) ☐ $V \rightarrow T^*$
- f) ☐ Nenhuma das anteriores

5) Com relação ao analisador sintático descendente recursivo, marque a opção que apresenta as afirmativas que são verdadeiras. (1 pt)

- I. Exige que a gramática esteja fatorada a esquerda
- II. Não apresenta retrocesso (*backtraking*)
- III. É um método ad hoc
- IV. É um método do tipo *bottom-up*
- V. Não é um tipo de analisador preditivo

Lucas Luan Pereira

- a) () I e II
- b) () I e III
- c) () II e III
- d) () II e IV
- * e) (~~X~~) I, II e III ✓
- f) () I, II e IV
- g) () II, III e IV
- h) () II, III, IV e V

6) Com relação a função *parse*, marque a opção que apresenta as afirmativas que são verdadeiras. (1 pt)

- I. Chama a função *gettoken* que por sua vez retorna um vetor de estruturas *tokentype*.
- II. É chamada pela função *match* para verificar se um código está sintaticamente correto.
- III. Tem como um de seus objetivos construir uma árvore sintática que representa o código fonte.
- IV. É uma função *void*
- V. Recebe como argumento um conjunto de *tokens* que representam um código fonte.

- a) () I
- * b) () II
- * c) (~~X~~) III ✓
- d) () I e II
- e) () I, IV
- f) () I, II e IV
- g) () I, III e IV
- h) () III, IV e V