

	Segunda Avaliação		Nota.
			10,0
Curso:	Ciência da Computação		
Disciplina:	Compiladores		
Aluno(a):		Data:	25/11/25

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

I. Exige que a gramática esteja fatorada a esquerda *OK*

II. Exige que a gramática apresente recursão à esquerda *X*

III. Não apresenta retrocesso (*backtracking*) *OK*

IV. Exige que para os símbolos variáveis com mais de uma regra de produção, os primeiros terminais deriváveis sejam capazes de identificar univocamente a produção que deve ser aplicada a cada instante da análise. *OK*

a) ( ) I e III

b) ( ) I e IV

c) ( ) II e III

d) ( ) II e IV

e) ( ) I, II, III

f) ( ) I, II e IV

*e* g) ~~( )~~ I, III e IV *!*

h) ( ) II, III, IV

i) ( ) I, II, III, IV

2) 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)

*e* a) ~~( )~~  $V \rightarrow (V \cup T)^*$  *!*

b) ( )  $V^* \rightarrow (V \cup T)^*$  *X*

c) ( )  $(V \cup T) \rightarrow (V \cup T)^*$  *^*

d) ( )  $V \rightarrow V^*$  *\*

e) ( )  $V \rightarrow T^*$

f) ( ) Nenhuma das anteriores

Considerando o seguinte código, referente a um nó de árvore sintática da linguagem Tiny responda as questões 3 e 4.

```
typedef enum { StmtK, ExpK } NodeKind;
typedef enum { IfK, RepeatK, AssignK, ReadK, WriteK } StmtKind;
typedef enum { OpK, ConstK, IdK } ExpKind;

typedef enum { Void, Integer, Boolean } ExpType;

#define MAXCHILDREN 3

typedef struct treeNode
{
    struct treeNode * child[MAXCHILDREN];
    struct treeNode * sibling;
    int lineno;
    NodeKind nodekind;
    union { StmtKind stmt; ExpKind exp; } kind;
    union { TokenType op;
            int val;
            char * name; } attr;
    ExpType type;
} TreeNode;
```

3) Marque a opção verdadeira com relação a variável *sibling*. (0,5 pt)

- ☐ a) É um ponteiro para o nó filho
- ☒ b) É um ponteiro para o nó irmão
- ☐ c) É um ponteiro para o lexema do *token* corrente.
- ☐ d) É um ponteiro para a lista de erros sintáticos da linha a qual se refere o nó.
- ☐ e) É um ponteiro para a árvore sintática construída pelo analisador sintático.
- ☐ f) Nenhuma das respostas anteriores

4) Marque a opção verdadeira com relação a variável *val*. (0,5 pt)

- ☐ a) Armazena o lexema do token corrente, caso a variável "kind" assuma o valor "IdK".
- ☐ b) Armazena o lexema do token ao qual se refere o nó.
- ☐ c) Armazena o valor resultante do cálculo de uma expressão aritmética, no caso de nós de operadores (+, -, \*, /, <).
- ☐ d) Armazena o valor resultante da avaliação da expressão lógica de controle de um *if* ou *repeat*.
- ☒ e) Nenhuma das respostas anteriores.

5) Marque a opção verdadeira com relação a tabela de símbolos (0,5 pt)

- ☐ a) Armazena os operadores da linguagem, por exemplo os aritméticos e os relacionais.
- ☐ b) Tem a função de armazenar o número da linha no código fonte na qual uma variável é declarada.
- ☐ c) É a principal estrutura de dados do analisador sintático.
- ☐ d) Armazena as palavras reservadas da linguagem.
- ☒ e) Nenhuma das respostas anteriores.



6) Com relação a função *parse* do analisador sintático da linguagem Tiny, 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) ☒ III

d) ☐ I e II

e) ☐ I, IV

f) ☐ I, II e IV

g) ☐ I, III e IV

h) ☐ III, IV e V

7) 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 *token* retornado pelo analisador léxico, se seu lexema é válido e emite uma mensagem de erro caso não seja.

c) ☐ Verifica para cada *token* retornado pelo analisador léxico se é uma marca válida 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 da análise sintática.

e) ☒ Emite uma mensagem de erro caso o *token* retornado pelo analisador léxico não coincida com o *token* esperado.