Universidade do Porto, Faculdade de Engenharia (FEUP)

Mestrado Integrado em Engenharia Informática e de Computadores

EIC0028-2S – COMPILADORES

Teste - 11 de Abril de 2012 *Duração total (I + II): 1 hora e 30 minutos*

Nota: Na parte I as respostas erradas têm cotação negativa. Numa pergunta com 4 alternativas, uma resposta errada tem uma cotação negativa igual a 1/3 da cotação da pergunta. Numa pergunta com 2 alternativas, a resposta errada tem uma cotação negativa igual à cotação da pergunta. As perguntas não respondidas têm cotação 0.

Nome: _____ Número: _____

PARTE I (9 valores)

1. Análise Lexical

1.1 [0,5 val] Os lexemas das linguagens de programação são especificados por expressões regulares devido ao facto de:

- ☐ não poderem ser representados por gramáticas sem contexto.
- ☐ as linguagens desses lexemas serem normalmente linguagens sem contexto e por isso a melhor forma de os representar é utilizando expressões regulares.
- ☒ as linguagens desses lexemas serem normalmente linguagens regulares e por isso a melhor forma de os representar é utilizando expressões regulares.
- ☐ as expressões regulares serem concisas e facilmente interpretadas pelos programadores dos compiladores.

1.2 [0,5val] Indique qual a sequência válida de *tokens* resultante da análise lexical de um compilador de código Java ao trecho de código: if2 = a * for;

- ☐ IDENTIFIER(“if2”) ASSIGN IDENTIFIER(“a”) MULT IDENTIFIER(“for”) SEMICOLON
- ☒ IDENTIFIER(“if2”) ASSIGN IDENTIFIER(“a”) MULT FOR SEMICOLON
- ☐ IF LITERAL(2) ASSIGN IDENTIFIER(“a”) MULT IDENTIFIER(“for”) SEMICOLON
- ☐ LHS(“if2”) STATEMENT(“=”)RHS(“a * for”)

1.3 [0,5val] Para representar os identificadores de linguagens de programação como C ou Java optaria por:

- ☐ Utilizar um autómato de pilha (PDA).
- ☐ Utilizar regras gramaticais que definem os identificadores.
- ☒ Utilizar uma expressão regular.
- ☐ Nenhuma das soluções indicadas nas outras alíneas.

2. Análise Sintática

2.1 [0,5val] Os analisadores sintáticos descendentes (também conhecidos por preditivos):

- ☐ Não permitem a geração da árvore sintática concreta.
- ☐ Precisam da utilização de autómatos de pilha (PDAs) para evitar certos conflitos gramaticais.
- ☐ Conseguem implementar todas as gramáticas livres de contexto.
- ☒ São uma das formas de implementar as gramáticas das linguagens de programação como C ou Java.

2.2 [0,5val] Considere a seguinte gramática livre do contexto com símbolos terminais a,b,c,d, e símbolos não terminais A e B, sendo A a variável inicial da gramática:

A → B A B | a **B → C | b | c | ε** **C → C d | c**

Qual das seguintes frases não faz parte da linguagem especificada pela gramática em questão?

- ☐ cddabcbb
- ☒ bcbcdddabd
- ☐ bcda
- ☐ cbbbacb

2.3 [0,5val] Na maioria dos compiladores, o objetivo da análise sintática é:

- ☐ Apenas identificar os erros gramaticais e dar as mensagens apropriadas.
- ☒ Identificar os possíveis erros sintáticos e gerar uma árvore que representa o programa a compilar.
- ☐ Identificar os possíveis erros lexicais e sintáticos.
- ☐ Identificar os possíveis erros sintáticos e gerar uma representação intermédia que inclui a conversão de tipos de dados.

3. Análise Semântica

3.1 [0,5val] Considere o pedaço de código Java: *int a, b; float d; double c; d = a*b*c;* Indique qual das seguintes respostas reflecte o resultado da análise semântica sobre este pedaço de código (considere que *d2i* realiza a conversão entre *double* e *int, i2d* entre *int* e *double, d2f* entre *double* e *float*):

- ☐ Uma representação intermédia que representa a expressão: d=a*d2i(i2d(b)*c);
- ☐ Uma representação intermédia que representa a expressão: d=d2f(i2d(a*b)*c);
- ☐ Uma representação intermédia que representa a expressão: d=i2d(a*b)*c;
- ☒ Reporta um erro semântico.

3.2 [1val] Indique a opção mais correcta:

- ☒ Num compilador a análise semântica altera as representações intermédias de acordo com as conversões de tipos de dados a efetuar.
- ☐ Num compilador a análise semântica tem de ser realizada utilizando a árvore sintática abstrata.
- ☐ Num compilador a análise semântica é apenas para avisar o programador.
- ☐ Nenhuma das outras opções.

4. Questões transversais

4.1 [0,5val] Supondo que do trecho de código Java: if (a<3) b=a; é gerada a sequência IF IDENTIFIER("a") LESSTHAN INT(3) IDENTIFIER("b") ASSIGN IDENTIFIER("a") SEMICOLON, indique qual a etapa de um compilador responsável por ter gerado essa sequência:

- ☒ análise léxical.
- ☐ análise sintática.
- ☐ análise semântica.
- ☐ nenhuma das outras opções.

5. [4val] Indique se cada uma das seguintes afirmações é verdadeira ou falsa:

V	F	
☐	☒	A existência de recursividade à esquerda numa gramática não impõe problemas quando se considera a implementação com um analisador sintático descendente da mesma gramática.
☐	☒	As linguagens livres de contexto (*context free languages*) podem ser todas implementadas por analisadores sintático descendentes desde que se use um valor de *lookahead* suficientemente alto.
☐	☒	As etapas de análise lexical, sintática e semântica limitam-se a identificar erros, e quando não existem erros há uma etapa subsequente que origina uma representação intermédia.
☐	☒	Para impor as precedências das operações na árvore de sintaxe nunca é necessário modificar a gramática de expressões aritméticas.
☐	☒	A separação entre análise lexical e sintática é apenas utilizada para explicar os conceitos já que na prática elas são implementadas em conjunto.
☐	☒	As etapas de análise lexical e sintática são dependentes do processador alvo.
☐	☒	As gramáticas LL(1) são gramáticas que assumem que o processamento da entrada é realizado da esquerda para a direita e que a análise necessita de ver dois tokens (o atual e o próximo).
☐	☒	A representação intermédia de nível alto mantém as construções de controlo (e.g., for, if) mas já reflecte como as variáveis são guardadas em memória.
☐	☒	A representação intermédia de nível alto é sempre uma árvore sintáctica abstrata (AST).
☐	☒	A ambiguidade numa gramática pode ser sempre removida modificando a gramática.