Compiladores – Prova #3

Nome:

Matrícula: Data:

Observações:

(a) A prova é individual e sem consulta, sendo vedado o uso de calculadoras e de telefones celulares.

(b) A interpretação dos comandos das questões faz parte da avaliação.

(c) A nota da prova será igual a 0 (zero) caso o estudante consulte algum material durante a prova, ou receba ou ofereça qualquer ajuda a outro estudante durante a prova.

(d) As questões podem ser resolvidas a lápis ou à caneta. Entretando, a resposta final deve ser destacada de forma clara (circulada, sublinhada, reforçada, indicada, etc...) para que a questão seja devidamente corrigida.

(e) O grampo da prova não deve ser removido. Caso o grampo seja removido, a nota da prova será igual a 0 (zero).

Parte A

1. (7 pontos) **Complete a sentença**: Uma definição dirigida pela sintaxe é denominada uma gramática de atributos se cada produção $A \rightarrow \alpha$ está associada a um conjunto de regras semânticas da forma $b := f(c_1, c_2, \ldots, c_k)$, onde f é uma função sem efeitos colaterais, $c_1, c_2, \ldots, c_k$ são atributos pertencentes aos símbolos gramaticais da produção e vale apenas uma das duas alternativas:

(i) b é um atributo sintetizado de A; ou

(ii) b é um atributo herdado, pertencente a um dos símbolos do lado direito da produção.

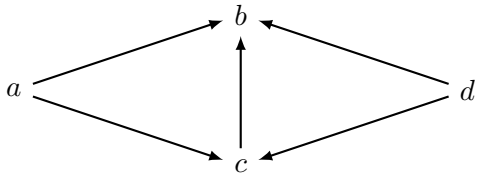
2. (5 pontos) Considere a esquema de tradução abaixo

$$A \rightarrow B \{ C.y := B.x \} C \{ A.z := B.x + C.y \}$$
$$B \rightarrow b \{ B.x := 1 \}$$
$$C \rightarrow c \{ C.w := 2 \}$$

Classifique os atributos dos não-terminais A, B e C como sintetizados ou herdados.

Solução: $A.z, B.x$ e $C.w$ são atributos sintetizados; $C.y$ é um atributo herdado.

3. (8 pontos) **Assinale a alternativa correta**. Considere o seguinte grafo de dependências:



Qual das ordenações dos atributos abaixo é topológica?

(A) $a - b - c - d$

(B) $b - c - d - a$

(C) $c - b - d - a$

(X) $d - a - c - b$

4. (10 pontos) **Julgue os itens abaixo.** Em cada item, preencha os parêntesis com V (verdadeiro) ou F (falso).

(F) Analisadores sintáticos preditivos demandam retrocesso.

(V) Se $A \to \alpha \mid \beta$ é uma gramática $LL(1)$, então no máximo um dentre α e β pode derivar ϵ.

(F) Em um analisador gramatical não-recursivo, um erro acontece se o próximo símbolo da entrada é a, A está no topo da pilha e a entrada $M[A, a]$ da tabela sintática tem duas ou mais produções.

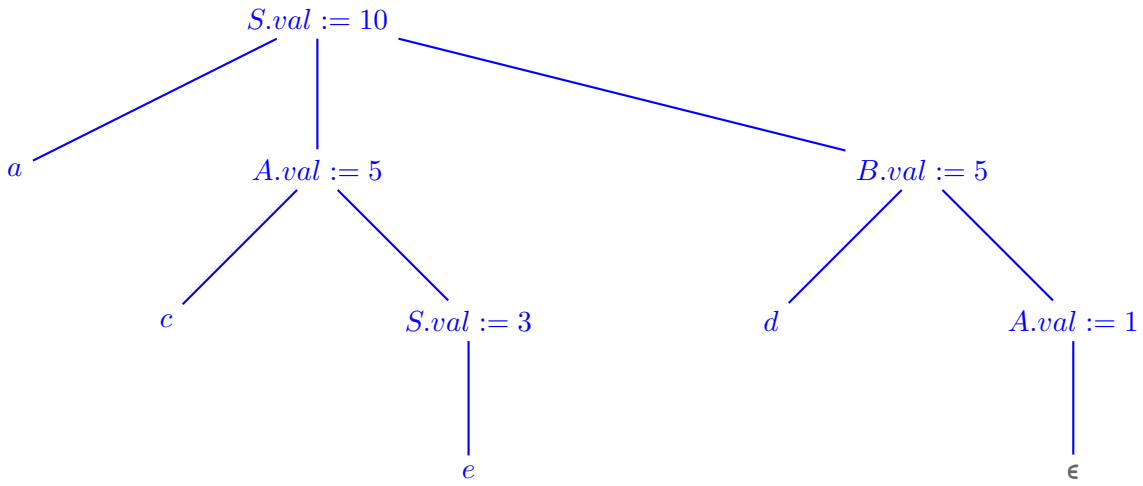
(F) Uma definição é denominada L-atribuída se todos os atributos são sintetizados.

(V) Uma restrição para o cálculo de atributos é que uma ação não pode referenciar um atributo sintetizado de um símbolo à direita da ação.

Parte B

5. (15 pontos) Para a expressão $aced$, construa uma árvore gramatical anotada de acordo com a definição dirigida pela sintaxe:

Produção	Regra semântica
$S \to aAB$	$S.val := A.val + B.val$
$S \to bBA$	$S.val := A.val \times B.val$
$S \to e$	$S.val := 3$
$A \to cS$	$A.val := 2 + S.val$
$A \to \epsilon$	$A.val := 1$
$B \to dA$	$B.val := A.val + 4$
$B \to \epsilon$	$B.val := 5$

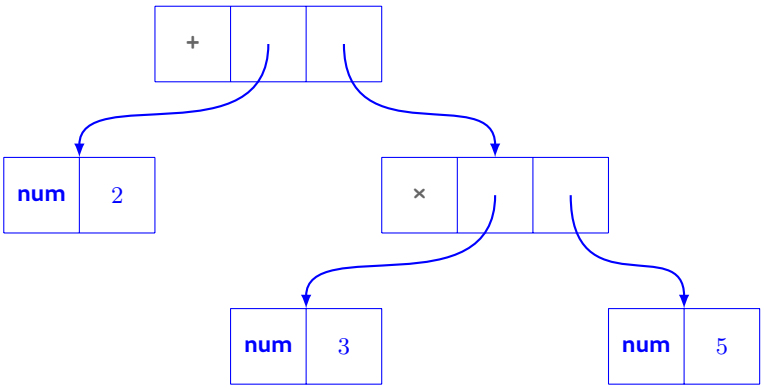


6. (15 pontos) A função criarNo(op, L, R) cria um nó de operador cujo rótulo é op, L é o ponteiro do operando à esquerda e R o ponteiro do operando à direita. A função criarFolha(**num**, val) cria um nó para um número, com rótulo **num**, cujo valor é indicado por val. O retorno de ambas funções é um ponteiro para o nó criado.

Determine uma sequência de chamadas das duas funções acima, com os devidos parâmetros e retornos, que construa a árvore sintática da expressão 2 + 3 × 5, assumindo que a multiplicação tem maior precedência em relação à adição.

Sequência de chamadas

$p_1 := \texttt{criarFolha}(\textbf{num}, 2)$
$p_2 := \texttt{criarFolha}(\textbf{num}, 3)$
$p_3 := \texttt{criarFolha}(\textbf{num}, 5)$
$p_4 := \texttt{criarNo}(\times, p_2, p_3)$
$p_5 := \texttt{criarNo}(+, p_1, p_4)$



7. Considere a gramática G abaixo:

$$S \to xR \mid Sy \mid xS \mid \epsilon$$
$$R \to Rx \mid zS \mid z$$

(a) (6 pontos) Determine os conjuntos primeiro( ) para todos os não-terminais de G.

$$\texttt{primeiro}(S) = \{\, x, \epsilon \,\}$$
$$\texttt{primeiro}(R) = \{\, z \,\}$$

(b) (6 pontos) Determine os conjuntos seguinte( ) para todos os não-terminais de G.

$$\texttt{seguinte}(S) = \{\, x, y, \$ \,\}$$
$$\texttt{seguinte}(R) = \{\, x, y, \$ \,\}$$

(c) (8 pontos) Construa a tabela sintática de um analisador predito não-recursivo para G.

Não-terminal	Símbolo da entrada			
	x	y	z	$\$$
S	$S \to xR$ $S \to Sy$ $S \to xS$ $S \to \epsilon$	$S \to Sy$ $S \to \epsilon$		$S \to Sy$ $S \to \epsilon$
R			$R \to Rx$ $R \to zS$ $R \to z$	

Parte C

8. Considere a gramática G abaixo:

$$A \to aA \mid bB \mid a$$
$$B \to Ba \mid Bb \mid c$$

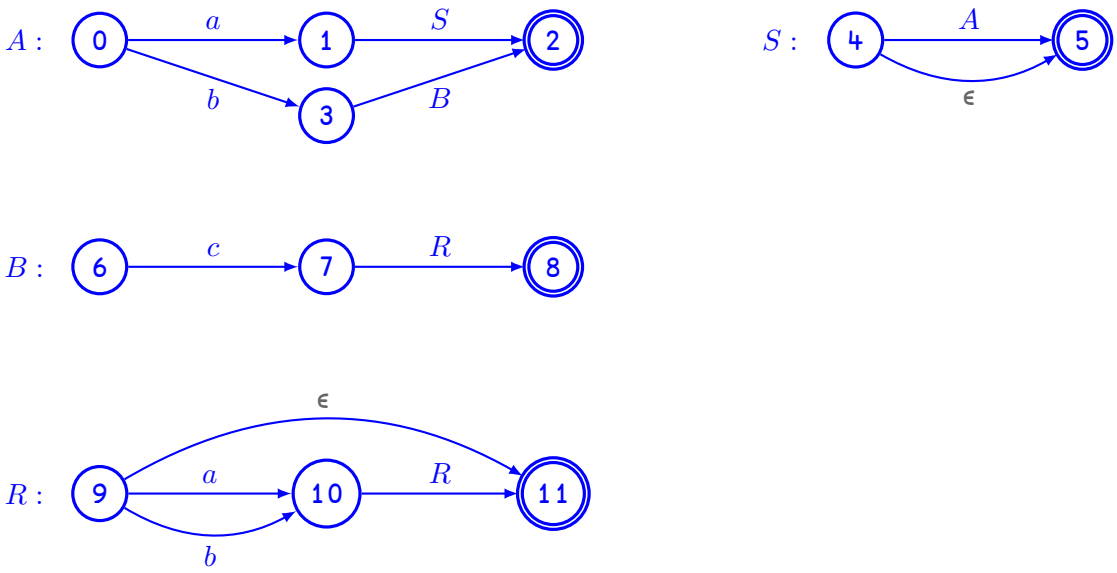
(a) (5 pontos) Determine a gramática G_1, obtida a partir da eliminação da recursão à esquerda de G.

$$A \to aA \mid bB \mid a$$
$$B \to cR$$
$$R \to aR \mid bR \mid \epsilon$$

(b) (5 pontos) Determine a gramática G_2, obtida a partir da aplicação de fatoração à esquerda em G_1.

$$A \to aS \mid bB$$
$$S \to A \mid \epsilon$$
$$B \to cR$$
$$R \to aR \mid bR \mid \epsilon$$

(c) (10 pontos) Gere, segundo o algoritmo apresentado em aula, os diagramas de transição de um analisador sintático preditivo recursivo para a gramática G_2.



(d) (15 pontos) Implemente em C, C++ ou Python, de acordo com os diagramas obtidos no item anterior, as funções associadas ao não-terminais de G_2 para
um analisador sintático preditivo recursivo. Assuma que `lookahead` já esteja definido e que contenha o próximo token da entrada e que as funções `reconhecer`(t) e `erro`( ) já estejam implementadas corretamente.
A implementação deve ser feita nas últimas páginas. Para fins de implementação, assuma que os terminais a, b, c sejam representados, no código, pelos caracteres ASCII `'a'`, `'b'`, `'c'`, respectivamente, e que o fim da entrada seja sinalizado pelo caractere `'$'`.

Solução em C:

```
void A()
{
    if (lookahead == 'a')
    {
        reconhecer('a');
        S();
    } else if (lookahed == 'b')
    {
        reconhecer('b');
        B();
    } else
```

```
        erro();
}

void S()
{
    if (lookahead != '$')
        A();
}

void B()
{
    reconhecer('c');
    R();
}

void R()
{
    if (lookahead == 'a' || lookahead == 'b')
    {
        reconhecer(lookahead);
        R();
    } else if (lookeahead != '$')
        erro();
}
```

Solução em Python:

```
def A():
    if lookahead == 'a':
        reconhecer('a')
        S()
    elif lookahed == 'b':
        reconhecer('b')
        B()
    else:
        erro()

def S():
    if lookahead != '$':
        A()

def B():
    reconhecer('c')
    R()

def R():
    if lookahead == 'a' or lookahead == 'b':
        reconhecer(lookahead)
        R()
    elif lookeahead != '$':
        erro()
```