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, ou alguma folha seja destacada da prova, a nota da prova será igual a 0 (zero).

Parte A

1. (5 pontos) **Complete a sentença**: Há ___três___ tipos gerais de analisadores sintáticos: os ___universais___, que podem tratar quaisquer gramáticas, mas que são ___ineficientes___ para um compilador de produção; os analisadores ___top-down___ e os analisadores ___bottom-up___.

2. (5 pontos) **Assinale a alternativa correta**. De acordo com as convenções de notação para gramáticas livres de contexto, letras maiúsculas do início do alfabeto representam

(A) terminais
(X) não-terminais
(C) cadeias de terminais
(D) símbolos gramaticais

3. (5 pontos) Considere a gramática livre de contexto abaixo:

$$S \to SaS \mid SbS \mid ab \mid ba \mid \epsilon$$

Determine uma derivação mais à esquerda para a cadeia $abab$ com, no máximo, 5 passos.

Solução:

$$\begin{aligned} S &\Rightarrow SaS \\ &\Rightarrow (ab)aS \\ &\Rightarrow aba(SbS) \\ &\Rightarrow aba(\epsilon)bS \\ &\Rightarrow abab(\epsilon) = abab \end{aligned}$$

4. (5 pontos) Identifique, na gramática abaixo, os terminais, os não-terminais e o símbolo de partida.

$$\begin{aligned} A &\to xB \mid yA \mid z \\ B &\to BA \mid AB \mid \epsilon \end{aligned}$$

Solução: A e B são não-terminais, x, y, z e ϵ são terminais e A é o símbolo de partida.

5. (10 pontos) **Julgue os itens abaixo.** Em cada item, preencha os parêntesis com V (verdadeiro) ou F (falso).

(V) Erros léxicos estão relacionados aos tipos e a identificação de tokens.

(V) Em relação à recuperação de erros, a modalidade de desespero tem a garantia de não entrar em um laço infinito.

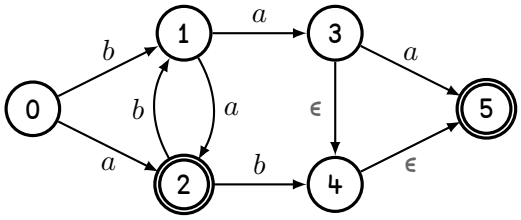
(V) Salvo indicação contrária, o lado esquerdo da primeira produção de uma gramática livre de contexto contém o símbolo de partida.

(F) Sejam α, β e γ três produções. Se $\alpha \stackrel{+}{\Rightarrow} \beta$ e $\beta \stackrel{*}{\Rightarrow} \gamma$ então $\alpha \stackrel{*}{\Rightarrow} \gamma$.

(F) Duas gramáticas G_1 e G_2 são equivalentes se têm o mesmo número de não-terminais.

Parte B

6. (10 pontos) Considere o grafo de transições do AFN abaixo.



Converta este AFN para uma gramática livre de contexto G.

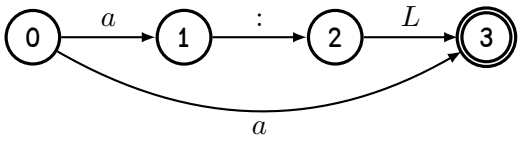
Solução:

$$
\begin{aligned}
A_0 &\to bA_1 \mid aA_2 \\
A_1 &\to aA_2 \mid aA_3 \\
A_2 &\to bA_1 \mid bA_4 \mid \epsilon \\
A_3 &\to A_4 \mid aA_5 \\
A_4 &\to A_5 \\
A_5 &\to \epsilon
\end{aligned}
$$

7. (10 pontos) Listas não-vazias de elementos do tipo a podem ser geradas pela gramática

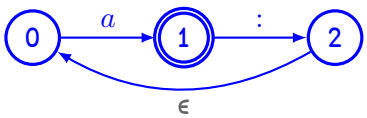
$$L \to a : L \mid a$$

cujo diagrama de transição é apresentado abaixo:

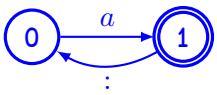


Simplifique este diagrama, gerando um novo diagrama equivalente e que tenha, no mínimo, um estado a menos.

Solução: Uma possível solução é eliminar a recursão, evitando a transição de 2 para 3, e tornar 1 um estado de aceitação:



A transição-ϵ também pode ser eliminada, de modo que o estado 2 também podem ser removido:



8. (15 pontos) Considere a gramática livre de contexto G abaixo:

$$D \rightarrow \textbf{unsigned int } I \mid \textbf{unsigned char } I \mid \textbf{int } I \mid \textbf{char } I$$
$$L \rightarrow a \mid b \mid \ldots \mid z \mid A \mid B \mid \ldots \mid Z$$
$$I \rightarrow LI \mid \epsilon$$

Gere uma gramática G', equivalente a G, por meio da aplicação da fatoração à esquerda.

Solução:

$$D \rightarrow \textbf{unsigned } D' \mid \textbf{int } I \mid \textbf{char } I$$
$$D' \rightarrow \textbf{int } I \mid \textbf{char } I$$
$$L \rightarrow a \mid b \mid \ldots \mid z \mid A \mid B \mid \ldots \mid Z$$
$$I \rightarrow LI \mid \epsilon$$

9. (20 pontos) Considere a gramática livre de contexto G abaixo:

$$S \rightarrow aS \mid bB \mid Aa$$
$$A \rightarrow SB \mid BA \mid c$$
$$B \rightarrow Sc \mid Ab$$

Gere uma gramática G', equivalente a G, por meio da aplicação da eliminação de recursão à esquerda.

Solução: Aplicando o algoritmo de eliminação de recursão à esquerda a partir da ordenação S, A, B dos não-terminais, no primeiro passo S não tem recursão simples à esquerda, resultando em

$$S \rightarrow aS \mid bB \mid Aa$$

No segundo passo, as produções-S devem ser substituídas em A, resultando em:

$$A \rightarrow aSB \mid bBB \mid AaB \mid BA \mid c$$

Como A agora tem recursão simples à esquerda, a eliminação da mesma produz

$$S \rightarrow aS \mid bB \mid Aa$$
$$A \rightarrow aSBA' \mid bBBA' \mid BAA' \mid cA'$$
$$A' \rightarrow aBA' \mid \epsilon$$

Por fim, substituíndo as produções-S e as produções-A em B obtemos

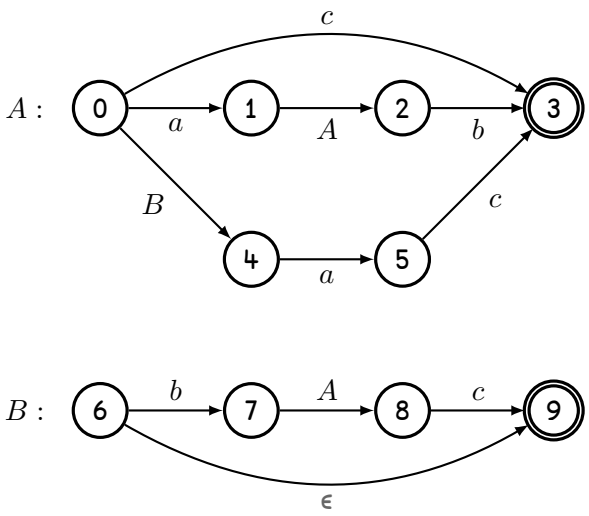
$$B \rightarrow aSc \mid bBc \mid Aac \mid aSBA'b \mid bBBA'b \mid BAA'b \mid cA'b$$

Eliminando a recursão simples à esquerda de B obtemos G':

$$S \rightarrow aS \mid bB \mid Aa$$
$$A \rightarrow aSBA' \mid bBBA' \mid BAA' \mid cA'$$
$$A' \rightarrow aBA' \mid \epsilon$$
$$B \rightarrow aScB' \mid bBcB' \mid AacB' \mid aSBA'bB' \mid bBBA'bB' \mid cA'bB'$$
$$B' \rightarrow AA'bB' \mid \epsilon$$

Parte C

10. (30 pontos) Considere os diagramas de transições da gramática livre de contexto G:



Implemente em C, C++ ou Python, de acordo com os diagramas apresentados, as funções associadas ao não-terminais de G para um analisador sintático preditivo recursivo. Assuma que a variável `lookahead` já esteja definida, que ela 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'` e `'c'`, respectivamente.

Importante: Use apenas elementos básicos da linguagem, sem recorrer a bibliotecas externas ou expressões regulares. Escreva o código com letra legível, de forma organizada e clara, usando as folhas pautadas do final da prova. O código não deve exceder 50 linhas.

Solução em C/C++:

```
1 void A()
2 {
3     if (lookahead == 'a')
4     {
5         reconhecer('a');
6         A();
7         reconhecer('b');
8     } else if (lookahead == 'c')
9         reconhecer('c');
10    else
11    {
12        B();
13        reconhecer('a');
14        reconhecer('c');
15    }
16 }
17
18 void B()
19 {
20    if (lookahead == 'b')
21    {
22        reconhecer('b');
23        A();
24        reconhecer('c');
```

```
    }
}
```

Solução em Python:

```python
def A():
    if lookahead == 'a':
        reconhecer('a')
        A()
        reconhecer('b')
    elif lookahead == 'c':
        reconhecer('c')
    else:
        B()
        reconhecer('a')
        reconhecer('c')


def B():
    if lookahead == 'b':
        reconhecer('b')
        A()
        reconhecer('c')
```