Compiladores – Prova #2

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. (5 pontos) **Complete a sentença**: A bufferização consiste no uso de um ou mais vetores/buffers auxiliares, que permitem a leitura da entrada em blocos de modo que o analisador léxico leia os caracteres a partir destes vetores/buffers, que são atualizados e preenchidos à medida do necessário.

2. (5 pontos) **Assinale a alternativa correta**. Seja $\mathcal{L}$ a linguagem gerada pela expressão regular

$$ab? \mid b?a$$

Qual é o valor de $|\mathcal{L}|$?

(A) 2

(X) 3

(C) 4

(D) 8

3. (5 pontos) **Assinale a alternativa correta.** Quantos são os prefixos de uma cadeia s de tamanho 7?

(A) 6

(B) 7

(X) 8

(D) 29

4. (5 pontos) Dada a cadeia $s =$ `"aba"`, compute a cadeia s^3.

Solução: $s^3 =$ `"abaabaaba"`

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

(V) Um analisador léxico recebe como entrada o programa-fonte e produz uma sequência de tokens.

(F) O conjunto dos números inteiros é um alfabeto.

(F) Seja $\mathcal{L}$ uma linguagem. Então $\mathcal{L}^* = \mathcal{L}^+ \cup \mathcal{L}$.

(V) Em um AFD, se uma cadeia s é aceita, existe um único caminho, que inicia no estado inicial e termina em um dos estado de aceitação, cuja concatenação dos rótulos das arestas, na ordem da travessia do caminho, resulta em s.

(V) O AFD equivalente a um AFN, obtido pelo algoritmo de conversão, terá, no máximo, $O(2^N)$ estados, onde N é o número de estados do AFN.

Parte B

6. (10 pontos) Seja $\mathcal{L} = \{ a, b \}$ e $\mathcal{M} = \{ 0, 1 \}$. Determine a linguagem

$$\mathcal{N} = \mathcal{L}^2\mathcal{M} \cup \mathcal{M},$$

listando todas as suas cadeias.

Solução:

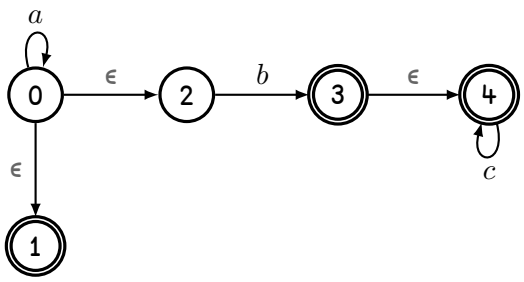
$$\mathcal{N} = \{ aa0, aa1, ab0, ab1, ba0, ba1, bb0, bb1, 0, 1 \}$$

7. (15 pontos) Defina uma expressão regular que denote todas as cadeias não-vazias formadas pelos caracteres a, b e c que não possuam duas vogais consecutivas. Por exemplo, as cadeias $a, abc, bac, bccb, \ldots$ devem ser denotadas por esta expressão regular.

Solução:

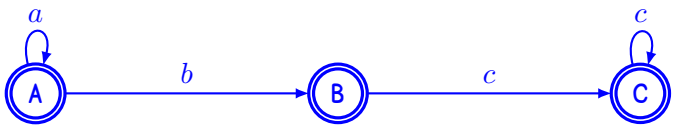
$$a?((b \mid c)a?)^+ \mid a$$

8. (15 pontos) Considere o grafo de transições do AFN abaixo.



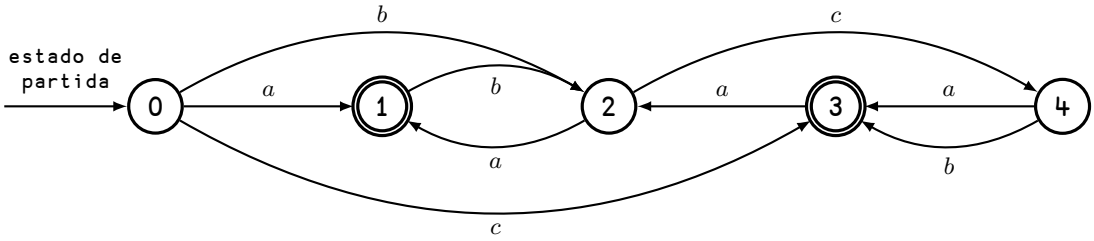
Usando o algoritmo apresentado em aula, converta este AFN em um AFD. **Atenção**: cada estado do AFD corresponde a um conjunto **não-vazio** de estados do AFN.

$$\begin{aligned}
&\texttt{Fechamento-}\epsilon(\texttt{0}) = \{\texttt{0, 1, 2}\} = \texttt{A}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(A, } a\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\texttt{0}\}) = \{\texttt{0, 1, 2}\} = \texttt{A}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(A, } b\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\texttt{3}\}) = \{\texttt{3, 4}\} = \texttt{B}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(A, } c\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\,\}) = \{\,\}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(B, } a\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\,\}) = \{\,\}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(B, } b\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\,\}) = \{\,\}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(B, } c\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\texttt{4}\}) = \{\texttt{4}\} = \texttt{C}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(C, } a\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\,\}) = \{\,\}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(C, } b\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\,\}) = \{\,\}\\
&\texttt{Fechamento-}\epsilon(\texttt{Movimento(C, } c\texttt{)}) = \texttt{Fechamento-}\epsilon(\{\texttt{4}\}) = \{\texttt{4}\} = \texttt{C}
\end{aligned}$$



Parte C

9. (30 pontos) Considere o AFD abaixo:



Implemente, em C, C++ ou Python, uma função que receba uma string s de tamanho N como parâmetro e que retorne verdadeiro, se todos os N caracteres de s compõem uma cadeia válida da linguagem definida pelo AFD; ou falso, caso contrário. Use apenas elementos básicos da linguagem, sem recorrer a bibliotecas externas ou expressões regulares. Use o diagrama como guia: a função deve aceitar todas as cadeias que o diagrama aceita, e recusar todas as cadeias que o diagrama recusa.

Importante: Escreva o código com letra legível, de forma organizada e clara, usando as folhas pautadas do final da prova. Assuma que a função `erro()` já esteja implementada e disponível para uso. O código não deve exceder 50 linhas.

Solução em C/C++:

```
bool isValid(const string& s)
{
    int state = 0;

    for (auto c : s) {
        int last = state;

        switch (state) {
        case 0:
            if (c == 'a')
                state = 1;
            else if (c == 'b')
                state = 2;
            else if (c == 'c')
                state = 3;
        break;

        case 1:
            if (c == 'b')
                state = 2;
        break;

        case 2:
            if (c == 'a')
                state = 1;
            else if (c == 'c')
                state =  4;
```

```
        break;

        case 3:
            if (c == 'a')
                state =  2;
        break;

        case 4:
            if (c == 'a' or c == 'b')
                state = 3;
        }

        if (state == last)
            return false;
    }

    return state == 1 or state == 3;
}
```

Solução em Python:

```
def isValid(s):

    state = 0

    for c in s:
        last = state

        if state == 0:
            if c == 'a':
                state = 1
            elif c == 'b':
                state = 2
            elif c == 'c':
                state = 3
        elif state == 1 and c == 'b':
            state = 2
        elif state == 2:
            if c == 'a':
                state = 1
            elif c == 'c':
                state =  4
        elif state == 3 and c == 'a':
            state =  2
        elif state == 4 and c in "ab":
            state = 3

        if state == last:
            return False

    return state in [1, 3]
```