

## Exercícios

- 1) Crie uma gramática para gerar números reais.

Defina a 4-upla de elementos da gramática  $G = (V, T, P, S)$

Exemplos de strings desta linguagem:  $\{ 2.5, +2.5, -2.5, +2, -2, +0, 0, .5, -.5 \}$

- 2) Faça uma gramática para gerar todos os números romanos.

Veja alguns exemplos de números romanos e seus respectivos valores em números arábicos:

I = 1	VI = 6	XX = 20	LXX = 70	CCC = 300	DCCC = 800
II = 2	VII = 7	XXX = 30	LXXX = 80	CD = 400	CM = 900
III = 3	VIII = 8	XL = 40	XC = 90	D = 500	M = 1000
IV = 4	IX = 9	L = 50	C = 100	DC = 600	MM = 2000
V = 5	X = 10	LX = 60	CC = 200	DCC = 700	MMM = 3000

Obs.:

- o menor número romano é: I (= 1)
- o maior número romano possível é: MMMCMXCIX (=3999)

Como proposta inicial para este exercício, faça apenas para os números de I a XCIX (1 a 99).

Dica: crie uma variável gramatical para a parte das dezenas e outra para a parte das unidades.

- 3) Refaça a gramática dos números reais (exercício 1), agora como gramática regular.
- 4) Crie expressões regulares para representar cada uma das linguagens regulares descritas a seguir:
- Números romanos de 1 a 99.
  - Valores monetários em Reais. Possuem exatamente duas casas decimais depois da vírgula, e usam ponto como separador de milhar.  
Exemplos:  $\{ R\$2,35 ; R\$1.546,98 ; R\$1,00 ; R\$10.000.000.000,00 \}$
  - Strings que possuem uma quantidade par de a's sobre o alfabeto  $\Sigma = \{a,b,c\}$
  - Strings que possuem tamanho múltiplo de três, sobre o alfabeto  $\Sigma = \{a,b,c\}$
  - Strings com pelo menos um '1' sobre o alfabeto  $\Sigma = \{0,1\}$
  - Strings terminadas em '110' sobre o alfabeto  $\Sigma = \{0,1\}$
  - Strings contendo '00' sobre o alfabeto  $\Sigma = \{0,1\}$
  - Strings iniciadas e terminadas em '10' sobre o alfabeto  $\Sigma = \{0,1\}$
  - Strings formadas por uma sequência de a's seguida de uma sequência de b's e terminada por uma sequência de c's. É possível que cada uma das três sequências descritas não possuam caracteres, mas se eles ocorrerem devem estar na ordem: a's, b's e c's. A string vazia não pode ser aceita. Considere o alfabeto  $\Sigma = \{a,b,c\}$

5) (ENADE Eng de Comp, 2017)

Um compilador transforma uma linguagem, em geral textual, em outra linguagem. Um dos tipos de linguagens que um compilador pode transformar são as linguagens regulares, que podem ser descritas utilizando-se expressões regulares compostas por símbolos isolados agrupados com operadores \* e U e organizadas com auxílio de parênteses.

Nesse contexto, avalie as afirmações a seguir.

- I. A palavra 10010100 pertence à linguagem representada por  $(100^*)^*$ .
- II. A palavra 10010 pertence à linguagem representada por  $(1(10)^*0)^*$ .
- III. Existe somente uma expressão regular para representar uma linguagem regular.

É correto o que se afirma em

- A** I, apenas.
- B** II, apenas.
- C** I e III, apenas.
- D** II e III, apenas.
- E** I, II e III.

6) (POSCOMP, 2004).

Seja  $\Sigma = \{a, b\}$ . Uma expressão regular denotando a linguagem  $L = \{w \in \Sigma^* \text{ tal que toda ocorrência de "a" em } w \text{ é imediatamente seguida de "b"}\}$  é:

- (a)  $(a^*b)^*$
- (b)  $(b + ab)^*$
- (c)  $a^*b$
- (d)  $b + (ab)^*$
- (e)  $(ab)^*$

7) (POSCOMP, 2004).

As seguintes expressões regulares denotam as linguagens  $P$ ,  $Q$ ,  $L$  e  $R$ , respectivamente:  $(1 + 10)^*$ ,  $(0 + 01)^*$ ,  $(0 + 1)^*$ ,  $0(11)^* + 1(00)^*$ . Não se pode afirmar que:

- (a)  $P \cap Q \neq \emptyset$
- (b)  $P \cup Q \neq L$
- (c)  $P \cap Q = \{\epsilon\}$
- (d)  $(1 + 0)^* \setminus P = Q$
- (e)  $R \subset L \setminus (P \cup Q)$

8) (POSCOMP, 2009)

Seja o alfabeto  $\Sigma = \{a, b\}$  e a linguagem regular

$$L = \{ \omega \mid \omega \in \Sigma^* \text{ e o n}^\circ \text{ de a's em } \omega \text{ é par} \}.$$

Qual das expressões regulares abaixo gera essa linguagem?

- A)  $(a b^* a b^*)^*$
- B)  $((a a)^* | b^*)^*$
- C)  $(b^* | (a a)^* | b^*)^*$
- D)  $(b^* a b^* a b^*)^*$
- E)  $(a a | b)^*$

9) (POSCOMP, 2012)

**Assinale a alternativa que apresenta, corretamente, uma expressão regular que denota todas as strings de a's e b's que têm pelo menos dois b's consecutivos.**

- a)  $(a^* - bb)(a+ba)^*(a+b)^*$
- b)  $(a+ba)^*bb(ba+a)^*$
- c)  $(a+b)^*ba^*b(a+b)^*$
- d)  $(a+bb)^*(bb+a)^*$
- e)  $(a+ba)^*bb(a+b)^*$

10) (ENADE, 2014)

Considere as seguintes expressões regulares cujo alfabeto é  $\{a, b\}$ .

$$R1 = a(a \cup b)^*$$

$$R2 = b(a \cup b)^*$$

Se  $L(R)$  é a linguagem associada a uma expressão regular  $R$ , é correto afirmar que

- A**  $L(R1) = L(R2)$ .
- B**  $L(R2) = \{w \mid w \text{ termina com } b\}$ .
- C** existe um autômato finito determinístico cuja linguagem é igual a  $L(R1) \cup L(R2)$ .
- D** se  $R3$  é uma expressão regular tal que  $L(R3) = L(R1) \cap L(R2)$ , então  $L(R3)$  é uma linguagem infinita.
- E** um autômato finito não determinístico que reconheça  $L(R1) \cup L(R2)$  tem, pelo menos, quatro estados.

11) (ENADE, 2014)

Expressões regulares constituem formas sucintas de descrever linguagens regulares. Uma de suas aplicações é descrever padrões a serem procurados em um texto. As expressões regulares R1, R2, R3 e R4 a seguir utilizam a seguinte convenção: o fecho de Kleene é denotado por \* e a união é denotada pelo símbolo |.

- $R1 = a^*ba^*ba^*ba^*$
- $R2 = a^*(a|b)a(a|b)^*$
- $R3 = a^*ab^*a(a|b)$
- $R4 = (a|b)^*$

Em relação às linguagens definidas pelas expressões regulares apresentadas, conclui-se que a cadeia **abbb** está contida apenas nas linguagens definidas por

- A** R1 e R4.
- B** R2 e R3.
- C** R2 e R4.
- D** R1 e R3.
- E** R2, R3 e R4.

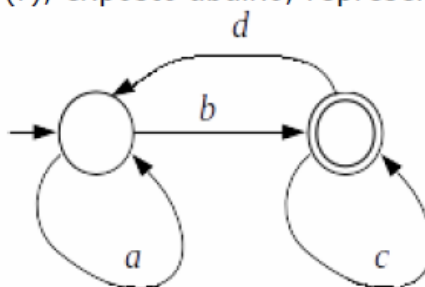
12) (POSCOMP, 2013)

Se o estado inicial for também estado final em um autômato finito, então esse autômato

- a) não aceita a cadeia vazia.
- b) não tem outros estados finais.
- c) é determinístico.
- d) aceita a cadeia vazia.
- e) é não determinístico.

13) (POSCOMP, 2016)

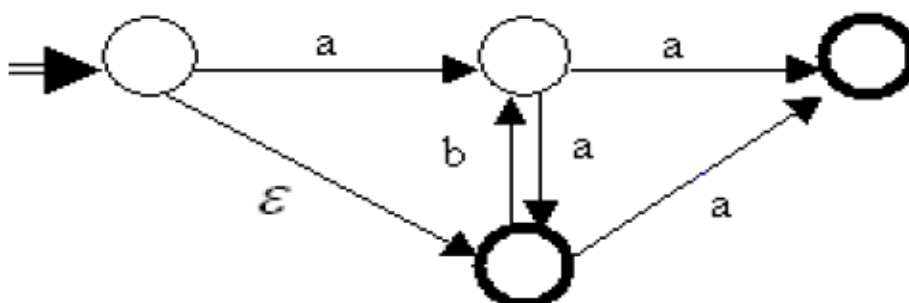
O grafo rotulado  $G(r)$ , exposto abaixo, representa qual expressão regular?



- A)  $r = ab^*(da^* + cb)^*$
- B)  $r = a^*b(d^* + cb)$
- C)  $r = (bb + d)^*(aa + c)^*$
- D)  $r = a^*b(c + da^*b)^*$
- E)  $r = a^*c^*(b + d)^*$

14) (POSCOMP, 2008)

Considere o autômato finito mostrado na figura abaixo (os círculos em negrito representam estados terminais).

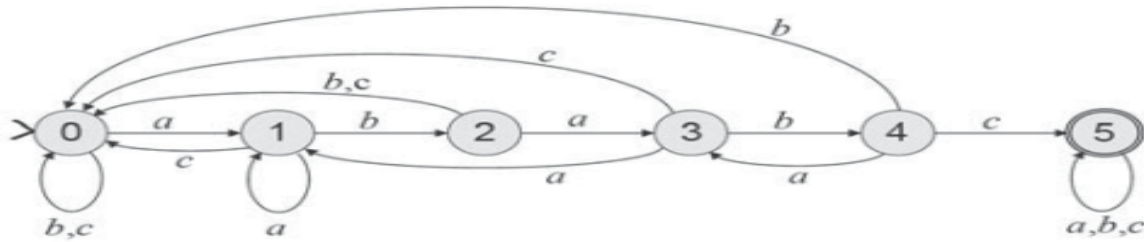


A esse respeito, assinale a afirmativa **FALSA**.

- A) A palavra *aaa* é reconhecida pelo autômato.
- B) A palavra *ababa* não é reconhecida pelo autômato.
- C) A palavra vazia é reconhecida pelo autômato.
- D) A palavra *aba* é reconhecida pelo autômato.
- E) A palavra *baba* é reconhecida pelo autômato.

15)(ENADE, 2011)

Autômatos finitos possuem diversas aplicações práticas, como na detecção de sequências de caracteres em um texto. A figura abaixo apresenta um autômato que reconhece sequências sobre o alfabeto  $\Sigma = \{a, b, c\}$  e uma gramática livre de contexto que gera um subconjunto de  $\Sigma^*$ , em que  $\lambda$  representa o *string* vazio.



$$S \rightarrow aS | bS | cS | abA$$

$$A \rightarrow abA | abcB$$

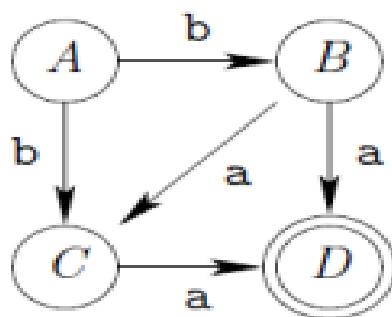
$$B \rightarrow aB | bB | cB | \lambda$$

Analizando a gramática e o autômato acima, conclui-se que

- A** a linguagem gerada pela gramática é inerentemente ambígua.
- B** a gramática é regular e gera uma linguagem livre de contexto.
- C** a linguagem reconhecida pelo autômato é a mesma gerada pela gramática.
- D** o autômato reconhece a linguagem sobre  $\Sigma$  em que os *strings* possuem o prefixo *ababc*.
- E** a linguagem reconhecida pelo autômato é a mesma que a representada pela expressão regular  $(a + b + c)^*(ab)^*abc(a + b + c)^*$ .

16) (POSCOMP, 2009)

Considere o autômato finito não-determinístico a seguir, sendo A o estado inicial e D o único estado de aceitação.

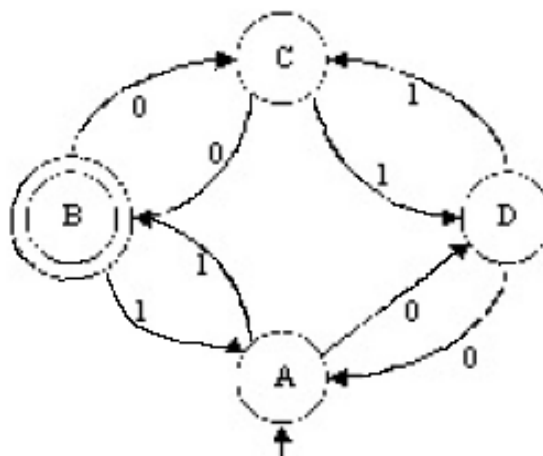


Que autômato finito determinístico com  $d$  como sua função de transição de estado aceita a mesma linguagem?

- A) Estado Inicial A, estados de aceitação C e D  
 $d(A, b) = B$   
 $d(B, a) = C$   
 $d(C, a) = D$
- B) Estado Inicial A, estado de aceitação C  
 $d(A, b) = B$   
 $d(B, a) = C$   
 $d(C, a) = C$
- C) Estado Inicial A, estado de aceitação D  
 $d(A, b) = B$   
 $d(B, a) = D$   
 $d(B, b) = C$   
 $d(C, a) = D$
- D) Todas as respostas acima estão corretas.
- E) É impossível converter esse autômato finito não determinístico em um autômato finito determinístico.

17) (ENADE, 2005)

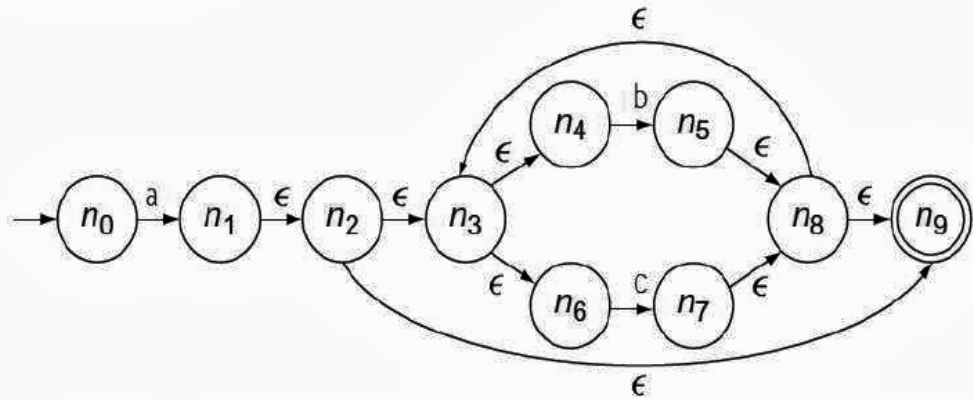
Que cadeia é reconhecida pelo autômato representado pelo diagrama de estados ao lado?



- A 101010
- B 111011000
- C 11111000
- D 10100
- E 00110011

18) (POSCOMP, 2012)

■ Considere o autômato a seguir.



(COOPER, K.; TORCZON, L. Engineering a Compiler. 2nd Edition. San Francisco: Morgan Kaufmann Publishers, 2012. p.51.)

**Assinale a alternativa que apresenta a expressão regular que gera a mesma linguagem reconhecida pelo autômato.**

- a)  $(ab)c^*$
- b)  $(a|b)c^*$
- c)  $a(b|c)^*$
- d)  $a(bc)^*$
- e)  $a(b)^*c$

19) Faça um AFD para cada item do exercício 4. Desenhe o grafo e a sua definição através da 5-upla de elementos  $M = (\Sigma, Q, \delta, q_0, F)$ .

Obs.: todo AFND ou  $AF\lambda$  gerado deve ser reconstruído como um AFD.

20) A partir do AFND fornecido através da quintupla de elementos, faça o desenho do grafo, converta-o para um AFD representando-o pelo desenho do grafo, reescreva a sua quintupla de elementos agora como AFD, escreva a ER equivalente e a gramática regular através de sua 4-upla de elementos.

$$M = (\Sigma, Q, \delta, q_0, F)$$

$$\Sigma = \{ a, b, c \}$$

$$Q = \{ q_0, q_1, q_2, q_3 \}$$

$\delta$  – função transição:

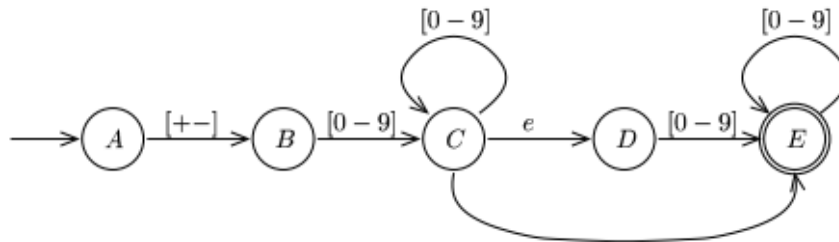
$\delta$	a	b	c
q0	{q0,q1}	{q0}	{q0}
q1	{}	{q2}	{}
q2	{}	{}	{q3}
q3	{}	{}	{}

$$F = \{q_3\}$$



21) (POSCOMP, 2002)

Assinale quantas seqüências de caracteres a seguir são reconhecidas pelo autômato finito abaixo. As quatro seqüências de caracteres (separadas por vírgulas) são: 0, +567, -89.5, -3e3.



(a) 0

(b) 1

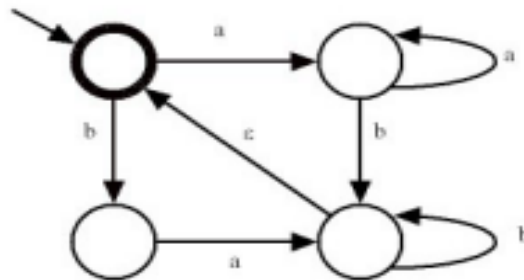
(c) 2

(d) 3

(e) 4

22) (POSCOMP, 2008)

Seja o autômato finito mostrado na figura abaixo que opera sobre o alfabeto  $\Sigma = \{a, b\}$  (o círculo em negrito indica um estado terminal):



Análise as seguintes afirmativas.

- I. O autômato finito mostrado na figura é determinístico.
- II. O autômato finito mostrado na figura é não-determinístico.
- III. O autômato finito mostrado na figura reconhece a palavra vazia.

A análise permite concluir que

- A) todas as afirmativas são falsas.
- B) somente a afirmativa I é falsa.
- C) somente a afirmativa II é falsa.
- D) somente a afirmativa III é falsa.
- E) nenhuma das afirmativas é falsa.

23) A partir do alfabeto  $\Sigma = \{a, b\}$ , faça um AFND e uma ER para reconhecer e denotar, respectivamente a linguagem:

$L = \{ w \mid \text{o quinto símbolo da direita para esquerda de } w \text{ é } a \}$

24) Para cada uma das linguagens regulares descritas a seguir, faça uma ER (expressão regular), em seguida faça um AFD (autômato finito determinístico), e por último escreva a GR (gramática regular) equivalente a partir do algoritmo de transformação de ADF em GR.

Considere as letras minúsculas  $\{a,b,c\}$  como alfabeto.

a.  $\{ a^i b^n a^p b^k \mid i \geq 0, n \geq 1, p \geq 2, k \geq 3 \}$

b.  $\{ a^i b a b^{2k} \mid i \geq 1, k \geq 0 \}$

c.  $\{ a^{2i} b^{2n+1} c^k \mid i,n,k \geq 0 \}$

d.  $\{ (ab)^i c b^k \mid i,k \geq 0 \}$

25) (POSCOMP, 2003)

Uma gramática  $G$  é definida por:

$$G = (\{x, y, z\}, \{S, W, X, Y, Z\}, P, S)$$

na qual os membros de  $P$  são:

$$\begin{aligned} S &\rightarrow WZ \\ W &\rightarrow X \mid Y \\ X &\rightarrow x \mid xX \\ Y &\rightarrow y \mid yY \\ Z &\rightarrow z \mid zZ \end{aligned}$$

Qual das expressões regulares abaixo corresponde a esta gramática?

(a)  $(xx^* \mid yy^*)zz^*$

(b)  $xx^* \mid yy^* \mid zz^*$

(c)  $xx^*(yy^* \mid zz^*)$

(d)  $(xx \mid yy)^*zz^*$

(e)  $xx^*yy^*zz^*$

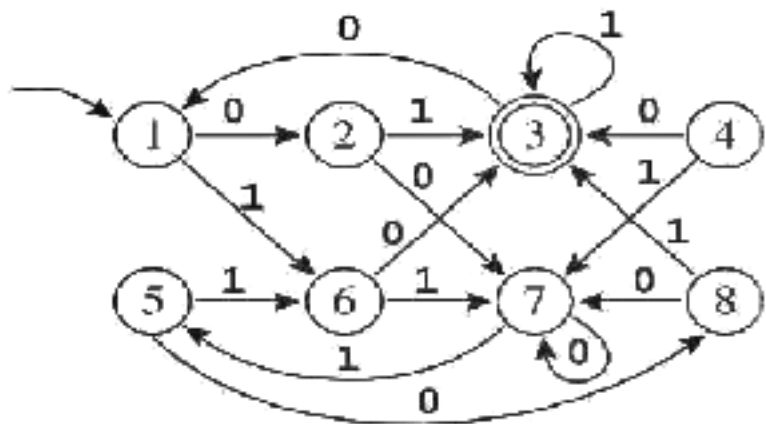
26) A partir de uma sequência de caracteres formada apenas por dígitos, ou seja, o alfabeto  $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ , faça um AFD para buscar o número 153.

27) Considerando um texto formado apenas por elementos do alfabeto  $\Sigma = \{a,b,c\}$ , construa uma AFD para buscar as palavras “aba” e “bac”.

28) Considerando um texto formado apenas por elementos do alfabeto  $\Sigma = \{0,1\}$ , construa uma AFD para buscar as seguintes três palavras: “10”, “111” e “01”.

29) (POSCOMP, 2013)

| Considere o autômato a seguir.



Sobre esse autômato, considere as afirmativas a seguir.

- I. Os estados 3 e 7 são equivalentes.
- II. Os estados 4 e 6 são equivalentes.
- III. Os estados 1 e 5 são equivalentes.
- IV. Os estados 2 e 8 são equivalentes.

Assinale a alternativa correta.

- a) Somente as afirmativas I e II são corretas.
- b) Somente as afirmativas I e IV são corretas.
- c) Somente as afirmativas III e IV são corretas.
- d) Somente as afirmativas I, II e III são corretas.
- e) Somente as afirmativas II, III e IV são corretas.

30) Verifique que o Lema do Bombeamento funciona nas seguintes linguagens regulares.

- a.  $a^* b c^+$
- b.  $(a|b)^* ab$
- c.  $(ab)^+$

31) Descubra se as seguintes linguagens são regulares ou não.

Dica: para provar que uma linguagem é regular basta apresentar uma das três abstrações (AFD, ER ou GR); para provar que ela não é regular use o Lema do Bombeamento provando por absurdo.

Observe ainda que apenas mostrar que uma linguagem satisfaz o lema do bombeamento não é suficiente para provar que ela é regular.

- a.  $\{ (ab)^n (ab)^n \mid n \geq 0 \}$
- b.  $\{ (ab)^n (cd)^n \mid n \geq 0 \}$
- c.  $\{ w \mid w \text{ é palavra da ling. com alfabeto } \{a,b\} \text{ e contém quantidade maior de símbolos 'b'} \}$

- d.  $\{ w \mid w \text{ é palavra da ling. com alfabeto } \{a,b\} \text{ e contém a mesma quantidade de 'a' e 'b' } \}$
- e.  $\{ (ab)^n a^k \mid n \geq k \}$
- f.  $\{ w \mid w \text{ é palavra da ling. com alfabeto } \{a,b\} \text{ e contém quantidade par de 'a' e par de 'b' } \}$
- g.  $\{ a^n b^k c^{n+k} \mid n, k \geq 0 \}$

32) (POSCOMP, 2013)

**| Sobre o Lema do Bombeamento (pumping lemma) para linguagens regulares, considere as afirmativas a seguir.**

- I. Se o alfabeto  $\Sigma = \{a, b\}$ , então pode-se provar por absurdo, por meio do Bombeamento, que a linguagem  $L_1 = \{w \in \Sigma^* \mid w \text{ termina com } b\}$  não é regular.
- II. Se o alfabeto  $\Sigma = \{a, b\}$ , então pode-se provar por absurdo, por meio do Bombeamento, que a linguagem  $L_2 = \{(a^n)^2 \mid n \geq 1\}$  não é regular.
- III. Se o alfabeto  $\Sigma = \{a, b\}$ , então pode-se provar por absurdo, por meio do Bombeamento, que a lin-

33) (POSCOMP, 2011)

Considere a seguinte propriedade sobre uma linguagem formal L: "Existe um número  $p \geq 0$ , tal que para qualquer palavra  $w \in L$ ,  $|w| \geq p$ , existem palavras  $x, y$  e  $z$ , com  $y \neq \epsilon$  e  $|xy| \leq p$ , tais que, para qualquer inteiro  $i \geq 0$ , a palavra  $xy^i z \in L$ ".

Com base no enunciado e nos conhecimentos sobre o tema, atribua V (verdadeiro) ou F (falso) para as afirmativas a seguir.

- ( ) Se L é aceita por AFND, então L satisfaz a propriedade acima.
- ( ) A linguagem formada de 1's e 0's com igual quantidade de ocorrências das palavras 01 e 10 satisfaz a propriedade acima.
- ( ) A propriedade acima é falsa para a linguagem  $0^i 1^k 2^j / i, j, k \geq 0$  e se  $i = 1$ , então  $k = j$ .
- ( ) A linguagem  $\{a^n b^n c^n / n \geq 0\}$  não satisfaz a propriedade acima.
- ( ) A linguagem  $\{a^n b^m / n, m \geq 0 \text{ e } n \neq m\}$  satisfaz a propriedade acima.

Assinale a alternativa que contém, de cima para baixo, a sequência correta.

- a) V, V, V, V, F.
- b) V, V, F, V, F.
- c) V, F, V, F, F.
- d) F, V, V, F, V.
- e) F, V, F, V, V.

34) (POSCOMP, 2012)

**Seja um Autômato Finito Não Determinístico (AFN) com 6 estados. Aplicando-se o algoritmo de conversão de um AFN para um Autômato Finito Determinístico (AFD), em quantos estados, no máximo, resultaria o AFD considerando-se os estados inúteis?**

- a) 12
- b) 36
- c) 64
- d) 1024
- e) 46656

35) Faça uma Máquina de Moore para classificar operadores lógicos AND, OR e NOT (maiúsculos ou minúsculos), variáveis formadas por letras minúsculas e a palavra reservada if (minúscula). Defina a sua 7-upla de elementos.

ER para cada saída da máquina:

```
<OP-LOGICO>      (a|A) (n|N) (d|D) | (o|O) (r|R) | (n|N) (o|O) (t|T)
<IF>              if
<VAR>             (a|...|z) +
```

Observe atentamente a característica determinística necessária para a máquina.

Exemplos de variáveis possíveis: a, an, anda, x, ifa

Observe ainda que a máquina não deve aceitar entradas como: Ax, Anda

36) Faça uma Máquina de Mealy sobre o alfabeto {0,1} para:

- a. Ligar (1) ou desligar (0) uma máquina. Como o início do funcionamento da máquina é custoso, o comando para ligar (1) deve ser repetido 3 vezes consecutivamente para que funcione. Já o comando de desligar (0) é dado uma única vez.
- b. Devolver o bit lido com atraso de uma transição. Para o primeiro bit emita um '0'.

Obs.: em todos os casos defina a sua 6-upla de elementos.

37) Faça uma máquina de estados finitos (decida entre Máquina de Moore e Máquina de Mealy) para representar a venda de café em um quiosque automatizado. O café custa 60 centavos e a máquina aceita somente moedas de 10, 50 centavos e 1 real. O café é feito automaticamente quando a máquina atinge crédito de 60 centavos. Havendo troco, o mesmo entra como crédito para o próximo café.

38) (POSCOMP, 2014)

Sobre as linguagens regulares, considere as afirmativas a seguir.

- I. As linguagens regulares podem ser expressas por máquinas de Moore e de Mealy.
- II. As linguagens regulares podem ser expressas por um autômato finito.
- III. Se  $A$  e  $B$  são linguagens regulares, então  $A \cap B$  também é.
- IV. Seja  $B = \{ba, na\}$ . Pode-se dizer que  $B^* = \{\lambda, ba, na, ab, an, babn, banan, naba, anab, nana, aban,$

39) (POSCOMP, 2004)

Seja a seguinte linguagem, onde  $\epsilon$  representa a sentença vazia:

S	→	AB		CD
A	→	a		$\epsilon$
B	→	b		f
C	→	c		g
D	→	h		i

Qual o conjunto de terminais que podem começar sentenças derivadas de S ?

- a) {a, c, g}
- b) {a, b, f, c, g}
- c) {a, b, f, c, g, h, i}
- d) {a, c, g, h, i}
- e) {a, b, f}

40) (POSCOMP, 2011)

Considere, a seguir, a gramática livre de contexto:

$$S \rightarrow aS | Sb | c$$

Qual expressão regular gera a mesma linguagem que a gramática definida acima?

- a)  $a^+ c a^+$
- b)  $a + b + c$
- c)  $a + c b +$
- d)  $c a^+ a^+$
- e)  $c a + b +$

41) (ENADE, 2008)

### DISCURSIVA

Qualquer expressão aritmética binária pode ser convertida em uma expressão totalmente parentizada, bastando reescrever cada subexpressão binária  $a \otimes b$  como  $(a \otimes b)$ , em que  $\otimes$  denota um operador binário. Expressões nesse formato podem ser definidas por regras de uma gramática livre de contexto, conforme apresentado a seguir. Nessa gramática, os símbolos não-terminais E, S, O e L representam expressões, subexpressões, operadores e literais, respectivamente, e os demais símbolos das regras são terminais.

$$E \rightarrow ( S O S )$$

$$S \rightarrow L | E$$

$$O \rightarrow + | - | * | /$$

$$L \rightarrow a | b | c | d | e$$

Tendo como referência as informações acima, faça o que se pede a seguir.

- A** Mostre que a expressão  $(a * (b / c))$  pode ser obtida por derivações das regras acima. Para isso, desenhe a árvore de análise sintática correspondente.
- B** Existem diferentes derivações para a expressão  $((a + b) * c) + (d * e)$ . É correto, então, afirmar que a gramática acima é ambígua? Justifique sua resposta.

42) Faça uma gramática livre de contexto (GLC) para cada uma das linguagens de nível 2, ou seja, livres de contexto.

- a.  $\{ a^{2i} b^{2i} c^{2k} \mid i \geq 0, k \geq 0 \}$
- b.  $\{ a^i b^i a^k b^k \mid i \geq 0, k \geq 1 \}$
- c.  $\{ a^i b^k a^k b^i \mid i, k \geq 0 \}$

- d.  $\{(ab)^i c b^i \mid i \geq 0\}$
- e.  $\{a^i c^k b^{2i} \mid i \geq 1, k \geq 0\}$
- f.  $\{a^i c^k b^n \mid i=k \text{ ou inclusivo } k=n\}$
- g.  $\{a^i b^n \mid i \text{ é diferente de } n\}$
- h.  $\{a^i b^n \mid 0 \leq i \leq n \leq 2i\}$
- i.  $\{a^{i+n} b^i c^n \mid i, n \geq 0\}$
- j. palíndromos ímpares sobre  $\{a,b\}$
- k. strings sobre  $\{a,b\}$  com a mesma quantidade de a's e b's.

43) (POSCOMP, 2015)

Considerando as linguagens  $L = \{0^n 1^n 2^i \mid n \geq 0 \text{ e } i \geq 0\}$  e  $M = \{0^i 1^n 2^n \mid n \geq 0 \text{ e } i \geq 0\}$ , pode-se afirmar que

- (A) a linguagem  $L \cup M$  pode ser gerada por uma gramática livre de contexto.
- (B) a linguagem  $M$  pode ser gerada por uma gramática regular.
- (C) a linguagem  $L$  pode ser aceita por um autômato finito determinístico.
- (D) a linguagem  $L \cap M$  pertence à classe das linguagens livres de contexto.
- (E) a linguagem  $M$  pode ser denotada por uma expressão regular.

44) (POSCOMP, 2010)

Considerando as linguagens  $L_1 = \{a^l c^m b^n; l \geq 0, m \geq 0, n \geq 0\}$  e  $L_2 = \{a^l c^m b^n; l \geq 0, m \geq 0, n = l + m\}$  sobre o alfabeto  $\Sigma = \{a, b, c\}$ , considere as afirmativas a seguir.

- I.  $L_1$  é uma linguagem regular.
- II.  $L_2$  é uma linguagem regular.
- III. Existe um autômato de pilha determinístico que reconhece  $L_1$ .
- IV. A linguagem  $L_2$  pode ser gerada pela  $G = (\{X, Y\}, \{a, b, c\}, \{X \rightarrow aXb, X \rightarrow Y, Y \rightarrow cYb, Y \rightarrow \lambda\}, X)$ , onde  $\lambda$  é a palavra vazia.

Assinale a alternativa correta.

- a) Somente as afirmativas I e II são corretas.
- b) Somente as afirmativas II e IV são corretas.
- c) Somente as afirmativas III e IV são corretas.
- d) Somente as afirmativas I, II e III são corretas.
- e) Somente as afirmativas I, III e IV são corretas.



Para a gramática a seguir, qual o conjunto de terminais que pode aparecer como primeiro terminal após o não-terminal A, em qualquer forma sentencial gerada pela gramática abaixo (isto é, não necessariamente imediatamente após A), onde  $\epsilon$  representa a sentença vazia?

$$S \rightarrow ABCDd$$

$$A \rightarrow aA \mid \epsilon$$

$$B \rightarrow bC \mid \epsilon$$

$$C \rightarrow cD \mid \epsilon$$

$$D \rightarrow e$$

- a) {d}
- b) {b}
- c) {b,c,e}
- d) {b,c,d,e}
- e) {e}

46) A partir da gramática fornecida, descreva, com precisão a linguagem gerada. Se a linguagem for regular, use a notação de expressão regular para a representá-la. Considere letras maiúsculas como símbolos não terminais (variáveis) e letras minúsculas como símbolos terminais.

a) 
$$\begin{aligned} S &\rightarrow aS \mid bB \mid \lambda \\ B &\rightarrow cB \mid \lambda \end{aligned}$$

b) 
$$\begin{aligned} S &\rightarrow aSb \mid aAb \\ A &\rightarrow bAa \mid \lambda \end{aligned}$$

c) 
$$\begin{aligned} S &\rightarrow aSbb \mid A \\ A &\rightarrow cA \mid c \end{aligned}$$

d) 
$$\begin{aligned} S &\rightarrow AAASb \mid \lambda \\ A &\rightarrow a \mid \lambda \end{aligned}$$

e) 
$$\begin{aligned} S &\rightarrow aaSB \mid \lambda \\ B &\rightarrow bB \mid b \end{aligned}$$

47) (POSCOMP, 2003).

Considere o seguinte trecho de programa:

```
1. i := 1;
2. while i <= n do
    begin
3.     sum := sum + a[i];
4.     i := i + 1;
    end;
```

Considere que:

- $I$  representa a inicialização da variável  $i := 1$  na linha 1;
- $T$  representa o teste da linha 2;
- $A$  representa os comandos da linha 3;
- $P$  representa o incremento na linha 4.

Qual é a expressão regular que representa todas as seqüências de passos possíveis de serem executados por este trecho de programa?

- (a)  $I(TAP)^+$
- (b)  $I(TAP)^*$
- (c)  $IT^+A^*P^*$
- (d)  $IT(APT)^*$
- (e)  $IT(APT)^+$

48) Crie uma gramática livre de contexto para gerar uma lista de expressões aritméticas formadas apenas por números inteiros sem sinal de 20 a 99 e pelo operador de adição (as expressões são separadas por ponto e vírgula).

Assuma a existência dos seguintes tokens:

- <DÍGITO1> - representa exatamente um dígito de 0 a 9.
- <DÍGITO2> - representa exatamente um dígito de 2 a 9.
- <SOMA> - representa o caractere '+'.
- <PT-VIRG> - representa o caractere ','.

Observe um exemplo de string que deve ser gerada pela gramática:

23 + 44 + 99 ; 25 ; 51 + 20

49) (ENADE, 2008)

Considere a gramática  $G$  definida pelas regras de produção ao lado, em que os símbolos não-terminais são  $S$ ,  $A$  e  $B$ , e os símbolos terminais são  $a$  e  $b$ .

$S \rightarrow AB$
$AB \rightarrow AAB$
$A \rightarrow a$
$B \rightarrow b$

Com relação a essa gramática, é correto afirmar que

- A** a gramática  $G$  é ambígua.
- B** a gramática  $G$  é uma gramática livre de contexto.
- C** a cadeia  $aabbbb$  é gerada por essa gramática.
- D** é possível encontrar uma gramática regular equivalente a  $G$ .
- E** a gramática  $G$  gera a cadeia nula.

- 50) Elabore uma gramática livre de contexto para gerar expressões com parênteses, colchetes e chaves balanceadas, as quatro operações básicas, números e variáveis. Admita a existência de tokens para representar cada um dos elementos citados.

Observe exemplos de strings da linguagem:

```
[2*(4+3)], 100, {w-[3-5]+{4-(x-((y)))-1}-9}-8, valor+contador,  
([{(2)}]), [2-{x-5}]
```

- 51) A partir do exercício anterior, faça a seguinte modificação: parênteses, colchetes e chaves são opcionais, mas se eles aparecerem na mesma expressão eles devem obedecer a seguinte ordem: parênteses (mais internos), colchetes e chaves (mais externos).

Exemplos corretos:

```
[2*(4+3)], 100, {w-[3-5]+{4-(x-((y)))-1}-9}-8, valor+contador
```

Exemplos incorretos:

```
([{(2)}]), [2-{x-5}]
```

- 52) **Desafio:** no exercício anterior, somente permita a ocorrência de colchetes se houver parênteses internos, e chaves se houver colchetes.

- 53) Desenvolva uma gramática livre de contexto para gerar expressões aritméticas no estilo da linguagem LISP. Use apenas números inteiros sem sinal. Considere, para cada operação, exatamente dois operandos. Suponha a existência de tokens para os elementos que compõem as expressões.

Exemplos:       (+ 2 3)  
                 (- (\* 20 30) 4)  
                 (+ (+ 1 4) (+ 3 (+ 3 5)))

- 54) Elabore uma gramática para gerar declarações de classes em Java. Assuma que o corpo da classe já está pronto.

Observações:

- Antes da palavra reservada `class` é possível aparecer um modificador de acesso: `public`, `private` ou `protected`. É possível a ocorrência da palavra `abstract` (antes ou depois do modificador de acesso).
- Só pode haver uma classe `Pai` no caso de herança com `extends`, e vários `Pais` em herança com `implements` (herança múltipla). As cláusulas `extends` e `implements` devem aparecer nesta ordem.
- Classes `interface` só podem herdar de classes `interface` (devem usar a palavra `implements`). Neste caso, não possuem modificador de acesso.

Observe quatro exemplos de strings desta linguagem:

```
public abstract class MinhaClasse { corpo }  
public class MinhaClasse extends Pai1 implements Pai2,Pai3 {<corpo>}  
abstract private class MinhaClasse extends Pai { <corpo> }  
interface MinhaInterface { <corpo> }
```

- 55) Sobre ambigüidade em gramáticas livres de contexto, resolva as seguintes situações:

- a) Crie uma gramática não ambígua para resolver o problema do `else` flutuante. Considere que um `else` sempre pertence ao `if` mais próximo.  
Por exemplo, no caso abaixo (com indentação errada de forma proposital) o `else` pertence ao `if (y>8)`

Considere disponíveis todos os tokens necessários.

```
if (x>5)
    if (y>8)
        comando1;
else
    comando2;
```

- b)** Prove que a gramática a seguir é ambígua.  
Proponha uma gramática equivalente não ambígua.

```
exp    ->    exp <OPER-ADIT> exp
          | <ABRE-PAR> exp <FECHA-PAR>
          | <NUM>
```

56) (POSCOMP, 2009)

Considere uma produção pertencente a uma gramática  $G$  dada por:

$$L \rightarrow L a S \mid S$$

Assinale a alternativa abaixo que, substituindo essa produção, elimina a recursividade à esquerda criando uma gramática equivalente:

- A)  $L \rightarrow R S$   
 $R \rightarrow a S R \mid \varepsilon$
- B)  $L \rightarrow S R$   
 $R \rightarrow a S R \mid \varepsilon$
- C)  $L \rightarrow S R$   
 $R \rightarrow S a R \mid \varepsilon$
- D)  $L \rightarrow S a R$   
 $R \rightarrow S a R \mid \varepsilon$
- E)  $L \rightarrow R S$   
 $R \rightarrow a R S \mid \varepsilon$

57) Retire a recursividade à esquerda e fatore (quando for possível) os trechos de gramáticas a seguir.

- a)**  $lista \rightarrow \langle IDENT \rangle \langle VIRG \rangle lista \mid \langle IDENT \rangle$
- b)**  $comandos \rightarrow comandos comando \mid \langle FIM \rangle$   
 $comando \rightarrow \langle IDENT \rangle \langle ATRIB \rangle exp$   
 $\quad \quad \quad \mid \langle IDENT \rangle tipo$   
 $tipo \rightarrow \langle REAL \rangle \mid \langle INT \rangle$   
 $exp \rightarrow exp \langle OPERADOR \rangle exp$   
 $\quad \quad \quad \mid \langle ABRE-PAR \rangle exp \langle FECHA-PAR \rangle$   
 $\quad \quad \quad \mid \langle NUM \rangle$   
 $\quad \quad \quad \mid \langle IDENT \rangle$

- c)  $S \rightarrow aS \mid bS \mid cS \mid aA \mid aB \mid aC \mid bA \mid bB \mid bC$
- d)  $lista \rightarrow lista \text{ <VIRG> fim} \mid lista \text{ <PT-VIRG> fim} \mid fim$
- e)  $termo \rightarrow termo \text{ outro} \mid fator \text{ <OP>} \mid fator \text{ <OPCONT>}$
- f)  $S \rightarrow AaB \mid AaS \mid a$
- g)  $S \rightarrow abB \mid aB \mid abC \mid abBC \mid ag \mid g \mid h$
- h)  $A \rightarrow Aa \mid b \mid c \mid Ad$
- i)  $S \rightarrow Sab \mid SA \mid aA \mid aS \mid c$
- j)  $exp \rightarrow exp \text{ <OP>} exp \mid termo$   
 $termo \rightarrow \text{<ABRE-PAR>} exp \text{ <FECHA-PAR>} \mid \text{<NUM>} \mid \text{<IDENT>}$

58) Reescreva as gramáticas na Forma normal de Chomsky e na Forma Normal de Greibach. Considere as letras maiúsculas como símbolos não terminais e as minúsculas como símbolos terminais.

- a)  $S \rightarrow aS \mid Sb \mid c$
- b)  $E \rightarrow E \circ E \mid a E a \mid n$

59) A partir da gramática BNF do Pascal fornecida no anexo A desta lista de exercícios, responda os itens solicitados. Não use seu conhecimento sobre a linguagem Pascal. É importante ressaltar que muitos erros que um compilador acusa são provenientes do analisador semântico e que não poderiam ser detectados pela gramática usada na análise sintática.

- a) O comando `for` permite expressões com variáveis em seus limites?  
Exemplo: `for x := y+1 to a*b do ...`
- b) A gramática permite números reais nos valores de limite de um comando `for`?  
Exemplo: `for x:= 2.5 to 8.5 do ...`
- c) A gramática permite um `for` onde os valores inicial e final estão invertidos?  
Exemplo: `for x:= 10 to 5 do ...`
- d) A gramática permite uma chamada aninhada nos índices de um vetor?  
Exemplo: `Vetor[Valor[Codigo[x]]]`
- e) O ponto-e-vírgula antes do `else` é proibido, obrigatório ou facultativo?
- f) A gramática aceita divisão por zero em expressões aritméticas?
- g) Os operadores `DIV` e `MOD` podem ser usados para números reais?

**h)** É possível uma construção aninhada de BEGIN e END?

Se sim, analise como ficam as duas situações abaixo, ou seja, verifique se o ponto-e-vírgula é proibido, facultativo ou obrigatório:

Situação 1:

```
BEGIN
    BEGIN
        x := x + 1;
    END;
END;
```

Situação 2:

```
BEGIN
    BEGIN
        x := x + 1;
    END;
    y := y + 2;
END;
```

**i)** É possível uma construção aninhada de WHILE?

Por exemplo:

```
while Flag1 do
    while Flag2 do
        while Flag3 do
            x := x + 1;
```

**j)** A estrutura while permite expressões condicionais contendo atribuições, tal como é permitido na Linguagem C?

Exemplo:

```
while x:=x+1 do ...
```

**k)** A estrutura while permite expressões contendo chamadas de funções?

Exemplo:

```
while Verifica(x) do ...
```

60) Analise a gramática do Java que segue no anexo B. Observe que ela usa um o formato BNF estendido (EBNF), ou seja, o lado direito das regras usa os mesmos metasímbolos que são permitidos nas expressões regulares.

A seta é substituída por um sinal de igual e os terminais são representados entre apóstrofes, além disso a regra completa é terminada por um ponto-e-vírgula.

Responda cada item abaixo a partir da análise sobre esta gramática.

**a)** Quantas classes podem ser herdadas com extends e com implements?

A linha de código abaixo é válida?

```
class Monitor extends Aluno, Pessoa implements Funcionario, Cidadão
```

É possível trocar de posição as palavras extends e implements?

**b)** No sintaxe do Java atual, o uso das chaves é obrigatório nos comandos de tratamento de erros `catch` e `finally`, mesmo que exista apenas uma linha em seu corpo. Sendo que é possível vários `catch` para um mesmo `try`, mas apenas um `finally`.  
Como a gramática fornecida se comporta diante de tal situação?

Exemplo correto:

```
try {
    ...
}
catch(Erro1 e) {
    coisa1();
}
catch(Erro2 e) {
    coisa2();
}
finally {
    coisa3();
}
```

Exemplo incorreto:

```
try {
    ...
}
catch(Erro1 e)
    coisa1();
catch(Erro2 e)
    coisa2();
finally
    coisa3();
```

**c)** Na declaração de uma função em Pascal é possível usar a escrita de um mesmo tipo para vários argumentos:  
`function FazAlgo(arg1,arg2:Integer; arg3,arg4:Real): UmTipo;`  
Existe possibilidade de fazer algo parecido em Java?

**d)** Existem várias situações nesta gramática onde as regras poderiam ser simplesmente substituídas por expressões regulares, ou seja, poderiam ser meramente tokens. Cite alguns casos em que isto ocorre, fornecendo em seguida a respectiva expressão regular.

61) Refaça o exercício 53 (gramática livre de contexto para gerar expressões aritméticas no estilo da linguagem LISP) utilizando-se do recurso do formato BNF.

62) (ENADE, 2014)

Uma gramática livre do contexto (GLC) é um modelo computacional geralmente utilizado para definir linguagens de programação e, quando está de acordo com a Forma de Backus-Naur (BNF), permite que alguns operadores sejam utilizados no lado direito de suas produções, como o operador `|` (*pipe*) que indica seleção, o operador `[ ]` que indica que o operando em questão é opcional, e o operador `*` que indica repetição de 0 ou mais vezes.

Projetar um compilador para uma determinada linguagem envolve, entre outras coisas, especificar quais são os símbolos válidos nesta linguagem, bem como quais são as regras sintáticas que a definem.

A linguagem de programação Java é uma linguagem com suporte à orientação a objetos que não permite herança múltipla e que permite que uma classe implemente múltiplas interfaces. A seguir, exibem-se trechos de código sintaticamente válidos na linguagem Java.

Trecho 1:

```
class A extends B { }
```

Trecho 2:

```
class F implements C { }
```

Trecho 3:

```
class J extends A implements C, D { }
```

No trecho 1, cria-se uma classe chamada A que herda de uma classe chamada B. No trecho 2, cria-se uma classe chamada F que implementa uma interface chamada C. No trecho 3, cria-se uma classe chamada J que herda de uma classe chamada A e implementa duas interfaces, chamadas C e D.

63) Refaça o exercício 53 (gramática livre de contexto para gerar expressões aritméticas no estilo da linguagem LISP) utilizando-se do recurso do formato EBNF.

64) Faça uma gramática no formato EBNF para gerar declaração de arrays multidimensionais em Pascal.

Formato geral:

```
lista_de_identificadores : array [li1..ls1, li1..ls1,..., li1..ls1] of tipo;
```

Exemplo de uma string da linguagem:

```
x,y,z : array [0..10,50..500] of Elemento;
```

Escreva duas soluções: uma considerando a existência de tokens já classificados, e outra partindo do ‘zero’, ou seja usando os símbolos terminais diretamente na gramática.

65) Reescreva cada gramática a seguir no formato EBNF.

a)     lista -> <IDENT> | <IDENT> <VIRG> lista

b)     exp -> termo <ADD> exp  
         | termo <SUB> exp  
         | termo  
termo -> <NUM> | <IDENT>

c)     inicio        -> <BEGIN> corpo <END> <EOF>  
corpo            -> declaraVar comandos  
declaraVar -> <DECVAR> listaVar  
listaVar        -> <IDENT> | <IDENT> listaVar  
comandos        -> comando comandos | comando  
comando         -> atribuicao | se | enquanto | leia | exhibe

d)     declaraFunção -> <FUNCTION> <IDENT> parâmetro <DOIS-PT>  
                         <IDENT> <PT-VIRG> declaraVar corpoFunção  
parâmetro -> <ABRE-PAR> listaArgumentos <FECHA-PAR> | λ  
listaArgumentos -> argumento <PT-VIRG> ListaArgumentos | argumento  
argumento -> referência defVar  
referência -> <VAR> | λ  
defVar -> listaIdent <DOIS-PT> tipo  
tipo -> <IDENT>  
listaIdent -> <IDENT> <VIRG> listaIdent | <IDENT>  
declaraVar -> λ | <VAR> listaDeclara  
listaDeclara -> defVar <PT-VIRG> listaDeclara | defVar  
corpoFunção -> blocoComandos



- 66) A partir da linguagem descrita a seguir, crie todos os tokens necessários através de suas expressões regulares. Crie também a gramática livre de contexto no formato EBNF.

A seguinte linguagem descrita representa a Lógica Proposicional e trabalha com expressões proposicionais. Ela possui 4 comandos:

1. comando de atribuição:

```
variável := expressão
```

2. comando de exibição:

```
out expressão_1, expressão_2, ..., expressão_n
```

3. comando de entrada de dados:

```
in variável_1, variável_2, ..., variável_n
```

4. comando condicional:

```
if(expressão) {  
    comandos_1  
}  
else {  
    comandos_2  
}
```

Os comandos de atribuição, exibição e de entrada de dados devem ser terminados por ponto-e-vírgula.

Os nomes das variáveis são formadas por uma sequência de letras minúsculas e/ou dígitos. Mas, sempre devem iniciar por letra.

A linguagem é case-sensitive.

A expressão é formada por parênteses balanceados, valores lógicos, variáveis, operadores lógicos.

Os valores lógicos podem ser: 1, 0, "v", "f", "verdadeiro", "falso", "false", "true" (observe que em alguns casos exige-se aspas).

Os operadores podem ser:  $\vee$ ,  $\wedge$ ,  $'$ ,  $\rightarrow$ ,  $\leftrightarrow$

Exemplos de programas aceitos:

```
in a;  
out a;
```

```
in a, b, c;  
out a ^ b, a v b, a -> c;
```

```
in a, b, c;  
d := a ^ (b <-> c);  
if(d) {  
    out d ^ a;  
}  
else{  
    out d ^ c;
```

```
}
```

```
a := "v";  
b := "f";  
c := a -> b;  
out c, c ^ 1;
```

```
in a,b,c;  
d := (b' v a) ^ ((b <-> c) v (c' ^ a)')';  
if(a ^ (b <-> c)) {  
    e := 1;  
    f := 0;  
    out d ^ a;  
}  
else{  
    if(d) {  
        e := 0;  
    }  
    else{  
        f := 1;  
    }  
    out d v a;  
}  
out e ^ f;
```

```
out "v" ^ "f" -> "true" <-> "false";
```

67) (ENADE, 2014)

Considere que:

- <classdecl> é um não terminal cujo intuito é dar origem a declarações de classes;
- <classbody> é um não terminal cujo intuito é dar origem ao corpo de classes;
- os terminais aparecem entre aspas duplas;
- "id" é um símbolo que representa qualquer identificador válido, como nomes de classes ou variáveis.

A gramática que especifica uma linguagem que não permita herança múltipla e que implemente zero ou mais interfaces é

- A** <classdecl> "class" "id" [ "extends " ] "id" <classbody>
- B** <classdecl> "class" "id" ("extends " "id")\* <classbody>
- C** <classdecl> "class" "id" [ "extends " ] "id" ["implements" ( , | "id")\* ] <classbody>
- D** <classdecl> "class" "id" [ "extends " "id" ] ["implements" "id" (" " "id")\* ] <classbody>
- E** <classdecl> "class" "id" [ "extends " "id" ] "implements" "id" (" " "id")\* <classbody>

68) (POSCOMP, 2014)

Sobre o lema do bombeamento (*pumping lemma*) para linguagens regulares, considere as afirmativas a seguir.

- I. Seja o alfabeto  $\Sigma = \{a, b\}$ . Pode-se provar por absurdo, através do bombeamento, que a linguagem  $L_1 = \{w \in \Sigma^* \mid w \text{ termina com } b\}$  não é regular.

69) (POSCOMP, 2008)

Analise as seguintes afirmativas.

- I. Todo autômato finito não-determinístico pode ser simulado por um autômato finito determinístico.
- II. Todo autômato finito determinístico pode ser simulado por um autômato finito não-determinístico.
- III. Todo autômato finito não-determinístico pode ser simulado por um autômato de pilha determinístico.
- IV. Todo autômato de pilha determinístico pode ser simulado por um autômato finito não-determinístico.
- V. Todo autômato finito não-determinístico pode ser simulado por uma máquina de Turing determinística.

A análise permite concluir que estão **CORRETAS**

- A) apenas as afirmativas I, II, III e IV.
- B) apenas as afirmativas II, III e V.
- C) apenas as afirmativas I, II, III e V.
- D) apenas as afirmativas II e IV.
- E) apenas as afirmativas I, II e IV.

70) De acordo com a Hierarquia de Chomsky, marque (3) para linguagem regular, (2) para linguagem livre de contexto e, (1) para linguagem sensível ao contexto, (0) para linguagem enumerável recursivamente.

- a) ☐ Autômato finito com movimentos vazios.
- b) ☐ Gramáticas usadas na maioria das linguagens de programação conhecidas.

- c) ( ) Autômato finito determinístico.
- d) ( ) Autômato finito não determinístico.
- e) ( ) Expressão regular.
- f) ( ) Máquina de Turing.
- g) ( )  $\{ a^i b^i c^i ; i > 0 \}$
- h) ( ) Linguagens naturais.
- i) ( )  $\{ a^i b^n c^i ; i, n > 0 \}$
- j) ( ) Linguagem tipo 3
- k) ( ) Linguagem tipo 2
- l) ( ) Linguagem tipo 1
- m) ( ) Linguagem tipo 0

71) Complete as frases usando conceitos adequados no contexto da Hierarquia de Chomsky.

- a) ..... é um formalismo usado para denotar (representar) linguagens regulares.
- b) ..... é um formalismo gerador (ou axiomático) para linguagens regulares.
- c) ..... é um formalismo reconhecedor para linguagens regulares.
- d) ..... é um formalismo gerador (ou axiomático) para linguagens livres de contexto.
- e) ..... é um formalismo reconhecedor para linguagens livres de contexto.
- f) ..... é um formalismo gerador (ou axiomático) para linguagens sensíveis ao contexto.
- g) ..... é um formalismo reconhecedor para linguagens sensíveis ao contexto.
- h) ..... é um formalismo gerador (ou axiomático) para ling. enumeráveis recursivamente.
- i) ..... é um formalismo reconhecedor para linguagens enumeráveis recursivamente.

72) (POSCOMP, 2002)

Sobre a Hierarquia de Chomsky, podemos afirmar:

- (a) Uma linguagem que é recursivamente enumerável não pode ser uma linguagem regular
- (b) As linguagens livres de contexto e as linguagens sensíveis a contexto se excluem
- (c) Uma linguagem que não é regular é livre de contexto
- (d) As linguagens reconhecidas por autômatos a pilha são as linguagens regulares
- (e) Há linguagens que não são nem livres de contexto nem sensíveis a contexto

73) (POSCOMP, 2007)

Seja a linguagem formal  $L = \{a^n b^{2n} c, n \geq 0\}$ . Analise as seguintes assertivas.

- I.  $L$  é uma linguagem livre de contexto.
- II. A gramática  $G = (\{S, X\}, \{a, b, c\}, \{S \rightarrow Xc, X \rightarrow aXbb | \epsilon\}, S)$  gera a linguagem  $L$ .
- III.  $L$  não pode ser reconhecida por um autômato com pilha.

A análise permite concluir que estão **CORRETAS**

- (a) apenas as assertivas I e II.
- (b) apenas as assertivas I e III.
- (c) apenas as assertivas II e III.
- (d) todas as assertivas.
- (e) nenhuma das assertivas.

74) (ENADE, 2005)

estado	símbolo lido na fita	símbolo gravado na fita	direção	próximo estado
início	●	●	direita	0
0	0	1	direita	0
0	1	0	direita	0
0	△	△	esquerda	1
1	0	0	esquerda	1
1	1	1	esquerda	1
1	●	●	direita	parada

Na tabela acima, estão descritas as ações correspondentes a cada um dos quatro estados (início, 0, 1, parada) de uma máquina de Turing, que começa a operar no estado “início” processando símbolos do alfabeto  $\{0, 1, \bullet, \triangle\}$ , em que ‘△’ representa o espaço em branco. Considere que, no estado “início”, a fita a ser processada esteja com a cabeça de leitura/gravação na posição 1, conforme ilustrado a seguir.

1	2	3	4	5	6	7	8	9	10	11	...
●	0	1	1	0	1	△	△	△	△	△	...

Considerando essa situação, assinale a opção que indica corretamente a posição da cabeça de leitura/gravação e o conteúdo

75) Enquadre, dentro da Hierarquia de Chomsky, cada uma das linguagens a seguir como livre de contexto (L) ou (S) sensível ao contexto.

Após a linguagem ter sido contextualizada no nível apropriado, construa o formalismo gerador mais adequado (GLC - gramática livre de contexto, ou GSC - gramática sensível ao contexto), e desenhe a máquina abstrata referente ao formalismo reconhecedor mais adequado (AP - autômato com pilha, ou MT - máquina de Turing).

- a) ( )  $\{ c a^n c b^n c \mid n \geq 0 \}$
- b) ( )  $\{ a^n b^n \mid n \geq 2 \}$
- c) ( ) palíndromos ímpares sobre  $\{a,b\}$
- d) ( )  $\{ a^i b^n c^k \mid k = i+n \}$
- e) ( )  $\{ a^i b^n c^k \mid n = i+k \}$
- f) ( )  $\{ a^{2i} b^{2i} c^{2k} \mid i \geq 0, k \geq 0 \}$
- g) ( )  $\{ a^n b^n c^n d^n \mid n \geq 1 \}$
- h) ( )  $\{ a^{2n} b a^n a^{2n} \mid n \geq 0 \}$
- i) ( )  $\{ a^i b^i a^k b^k \mid i \geq 0, k \geq 1 \}$
- j) ( )  $\{ a^n b^k a^n b^k \mid i \geq 0, k \geq 0 \}$
- k) ( )  $\{ a^i b^k a^k b^i \mid i, k \geq 0 \}$
- l) ( )  $\{ (ab)^i c b^i \mid i \geq 0 \}$
- m) ( ) strings sobre  $\{a,b\}$  com a mesma quantidade de a's e b's.
- n) ( )  $\{ a^i b^n \mid i \text{ é diferente de } n \}$
- o) ( )  $\{ bb a^i bb a^i bb a^i bb \mid i \geq 1 \}$
- p) ( )  $\{ a^{2i} b^i c^{2i} \mid i \geq 0 \}$
- q) ( )  $\{ w c w \mid w \text{ é uma string qualquer sobre } \{a,b\} \}$
- r) ( )  $\{ w c w c w \mid w \text{ é uma string qualquer sobre } \{a,b\} \}$

76) (POSCOMP, 2015)

A gramática  $G = (\{S, A, B\}, \{0, 1\}, P, S)$ , onde  $P$  é dado pelas regras de produção

$S \rightarrow 0AB \mid 1BA$

$A \rightarrow 0AS \mid 1A \mid \varepsilon$

$B \rightarrow 0B \mid 1BS \mid \varepsilon$

gera uma linguagem que

- (A) pertence à classe Regular.
- (B) contém a cadeia vazia  $\varepsilon$ .
- (C) pode ser aceita por um autômato com pilha.
- (D) pode ser denotada por uma expressão regular.
- (E) é igual ao conjunto de cadeias  $\{x \in \{0, 1\}^* \mid x \text{ tem quantidade igual de zero (0) e de um (1)}\}$

77) (POSCOMP, 2012)

Sobre gramáticas e linguagens, considere as afirmativas a seguir.

- I. Uma gramática na Forma Normal de Chomsky pode ser ambígua.
- II. Uma gramática ambígua pode gerar uma linguagem inerentemente não ambígua.
- III. Uma gramática na Forma Normal de Greibach pode ser convertida para a Forma Normal de Chomsky.
- IV. O algoritmo de conversão de Gramática Livre de Contexto para Gramática na Forma Normal de Chomsky pode ser diretamente aplicado a uma gramática que não seja  $\lambda$ -livre.

Assinale a alternativa correta.

- a) Somente as afirmativas I e II são corretas.
- b) Somente as afirmativas I e IV são corretas.
- c) Somente as afirmativas III e IV são corretas.
- d) Somente as afirmativas I, II e III são corretas.
- e) Somente as afirmativas II, III e IV são corretas.

78) (POSCOMP, 2014)

Observe a gramática a seguir.

$S \rightarrow aAbba$

$aAb \rightarrow aabbbA \mid ab$

$bAb \rightarrow bbA$

$bAa \rightarrow Bbaa$

$bB \rightarrow Bb$

$aB \rightarrow aA$

Sobre essa gramática, assinale a alternativa correta.

- a) É irrestrita e aceita a linguagem  $\{a^n b^{2n+1} a^n \mid n \geq 1\}$ .
- b) É irrestrita e aceita a linguagem  $\{a^n b^{2n} a^n \mid n \geq 1\}$ .
- c) É sensível ao contexto e aceita a linguagem  $\{a^n b^{2n+1} a^n \mid n \geq 1\}$ .
- d) É sensível ao contexto e aceita a linguagem  $\{a^n b^{2n} a^n \mid n \geq 1\}$ .
- e) É livre de contexto e aceita a linguagem  $\{a^n b^{2n+1} a^n \mid n \geq 1\}$ .

79) (POSCOMP, 2016)

A linguagem  $L = \{a^n b^m \mid n \leq m + 3\}$ , para  $n \geq 0$  e  $m \geq 0$ , é:

- A) Regular e gerada pela gramática  $S \rightarrow aA, A \rightarrow baA \mid \varepsilon$ .
- B) Sensível ao contexto e gerada pela gramática  $S \rightarrow aSBC, S \rightarrow aBC, CB \rightarrow BC, aB \rightarrow ab, bB \rightarrow bb, bC \rightarrow bc, cC \rightarrow cc$ .
- C) Recursivamente enumerável e gerada por uma gramática sem restrições nas regras de produção.
- D) Estrutura de frase e gerada por uma gramática sem restrições nas regras de produção.
- E) Livre de contexto e gerada pela gramática  $S \rightarrow aaaA, A \rightarrow aAb \mid B, B \rightarrow Bb \mid \varepsilon$ .

80) (ENADE Cienc Comp, 2017)

Considere o seguinte alfabeto:

$\Sigma = \{ (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, - \}$ .

Considere, ainda, uma linguagem  $L$  definida sobre esse alfabeto.

$L = \{w \mid w \in \Sigma^*, \text{ para cada ocorrência de '(' em } w, \text{ existe uma ocorrência de ')'}\}$

Por exemplo, a cadeia  $x = (2 + (3 - 4))$  pertence a  $L$ , mas a cadeia  $y = (2 + (3 - 4)$  não pertence a  $L$ .

Com relação à linguagem  $L$ , avalie as asserções a seguir e a relação proposta entre elas.

- I. A linguagem  $L$  não pode ser considerada regular.

**PORQUE**

- II. Autômatos finitos não possuem mecanismos que permitam contar infinitamente o número de ocorrências de determinado símbolo em uma cadeia.

A respeito dessas asserções, assinale a opção correta.

- A** As asserções I e II são proposições verdadeiras, e a II é uma justificativa correta da I.
- B** As asserções I e II são proposições verdadeiras, mas a II não é uma justificativa correta da I.
- C** A asserção I é uma proposição verdadeira, e a II é uma proposição falsa.
- D** A asserção I é uma proposição falsa, e a II é uma proposição verdadeira.
- E** As asserções I e II são proposições falsas.



SUDKAMP, Thomas A. **Languages and machines**: an introduction to the theory of computer science. 2. ed. Massachusetts: Addison-Wesley Publishing Company, Inc., 1997. Appendix III.

## Backus-Naur Definition of Pascal

---

The programming language Pascal was developed by Niklaus Wirth in the late 1960s. The language was defined using the notation known as the Backus-Naur form (BNF). The metasymbol  $\{u\}$  denotes zero or more repetitions of the string inside the brackets. Thus the BNF rule  $A \rightarrow \{u\}$  is represented in a context-free grammar by the rules  $A \rightarrow uA \mid \lambda$ .

$\langle \text{program} \rangle \rightarrow \langle \text{program heading} \rangle ; \langle \text{program block} \rangle$

$\langle \text{program block} \rangle \rightarrow \langle \text{block} \rangle$

$\langle \text{program heading} \rangle \rightarrow \mathbf{program} \langle \text{identifier} \rangle (\langle \text{file identifier} \rangle \{, \langle \text{file identifier} \rangle \})$

$\langle \text{file identifier} \rangle \rightarrow \langle \text{identifier} \rangle$

$\langle \text{identifier} \rangle \rightarrow \langle \text{letter} \rangle \{ \langle \text{letter or digit} \rangle \}$

$\langle \text{letter or digit} \rangle \rightarrow \langle \text{letter} \rangle \mid \langle \text{digit} \rangle$

$\langle \text{letter} \rangle \rightarrow a \mid b \mid \dots \mid z$

$\langle \text{digit} \rangle \rightarrow 0 \mid 1 \mid \dots \mid 9$

$\langle \text{unpacked structured type} \rangle \rightarrow \langle \text{array type} \rangle \mid \langle \text{record type} \rangle \mid \langle \text{set type} \rangle \mid \langle \text{file type} \rangle$   
 $\langle \text{array type} \rangle \rightarrow \mathbf{array} [\langle \text{index type} \rangle \{, \langle \text{index type} \rangle\}] \mathbf{of} \langle \text{component type} \rangle$   
 $\langle \text{index type} \rangle \rightarrow \langle \text{simple type} \rangle$   
 $\langle \text{component type} \rangle \rightarrow \langle \text{type} \rangle$   
 $\langle \text{record type} \rangle \rightarrow \mathbf{record} \langle \text{field list} \rangle \mathbf{end}$   
 $\langle \text{field list} \rangle \rightarrow \langle \text{fixed part} \rangle \mid \langle \text{fixed part} \rangle ; \langle \text{variant part} \rangle \mid \langle \text{variant part} \rangle$   
 $\langle \text{fixed part} \rangle \rightarrow \langle \text{record section} \rangle [; \langle \text{record section} \rangle]$

$\langle \text{result type} \rangle \rightarrow \langle \text{type identifier} \rangle$   
 $\langle \text{statement part} \rangle \rightarrow \langle \text{compound statement} \rangle$   
 $\langle \text{statement} \rangle \rightarrow \langle \text{unlabeled statement} \rangle \mid \langle \text{label} \rangle : \langle \text{unlabeled statement} \rangle$   
 $\langle \text{unlabeled statement} \rangle \rightarrow \langle \text{simple statement} \rangle \mid \langle \text{structured statement} \rangle$   
 $\langle \text{simple statement} \rangle \rightarrow \langle \text{assignment statement} \rangle \mid \langle \text{procedure statement} \rangle \mid$   
 $\quad \langle \text{go to statement} \rangle \mid \langle \text{empty statement} \rangle$   
 $\langle \text{assignment statement} \rangle \rightarrow \langle \text{variable} \rangle := \langle \text{expression} \rangle \mid$   
 $\quad \langle \text{function identifier} \rangle := \langle \text{expression} \rangle$   
 $\langle \text{variable} \rangle \rightarrow \langle \text{entire variable} \rangle \mid \langle \text{component variable} \rangle \mid \langle \text{referenced variable} \rangle$   
 $\langle \text{entire variable} \rangle \rightarrow \langle \text{variable identifier} \rangle$   
 $\langle \text{variable identifier} \rangle \rightarrow \langle \text{identifier} \rangle$   
 $\langle \text{component variable} \rangle \rightarrow \langle \text{indexed variable} \rangle \mid \langle \text{field designator} \rangle \mid \langle \text{file buffer} \rangle$   
 $\langle \text{indexed variable} \rangle \rightarrow \langle \text{array variable} \rangle [ \langle \text{expression} \rangle \{ , \langle \text{expression} \rangle \} ]$   
 $\langle \text{array variable} \rangle \rightarrow \langle \text{variable} \rangle$

$\langle \text{set} \rangle \rightarrow [ \langle \text{element list} \rangle ]$   
 $\langle \text{element list} \rangle \rightarrow \langle \text{element} \rangle \{ , \langle \text{element} \rangle \} \mid \langle \text{empty} \rangle$   
 $\langle \text{element} \rangle \rightarrow \langle \text{expression} \rangle \mid \langle \text{expression} \rangle .. \langle \text{expression} \rangle$   
 $\langle \text{procedure statement} \rangle \rightarrow \langle \text{procedure identifier} \rangle \mid$   
 $\qquad \qquad \qquad \langle \text{procedure identifier} \rangle ( \langle \text{actual parameter} \rangle$   
 $\qquad \qquad \qquad \{ , \langle \text{actual parameter} \rangle \} )$   
 $\langle \text{procedure identifier} \rangle \rightarrow \langle \text{identifier} \rangle$   
 $\langle \text{actual parameter} \rangle \rightarrow \langle \text{expression} \rangle \mid \langle \text{variable} \rangle \mid$   
 $\qquad \qquad \qquad \langle \text{procedure identifier} \rangle \mid \langle \text{functional identifier} \rangle$   
 $\langle \text{go to statement} \rangle \rightarrow \textbf{goto} \langle \text{label} \rangle$   
 $\langle \text{empty statement} \rangle \rightarrow \langle \text{empty} \rangle$   
 $\langle \text{empty} \rangle \rightarrow \lambda$   
 $\langle \text{structured statement} \rangle \rightarrow \langle \text{compound statement} \rangle \mid \langle \text{conditional statement} \rangle \mid$   
 $\qquad \qquad \qquad \langle \text{repetitive statement} \rangle \mid \langle \text{with statement} \rangle$   
 $\langle \text{compound statement} \rangle \rightarrow \textbf{begin} \langle \text{statement} \rangle \{ ; \langle \text{statement} \rangle \} \textbf{end}$   
 $\langle \text{conditional statement} \rangle \rightarrow \langle \text{if statement} \rangle \mid \langle \text{case statement} \rangle$   
 $\langle \text{repetitive statement} \rangle \rightarrow \langle \text{for statement} \rangle \mid \langle \text{while statement} \rangle \mid$   
 $\qquad \qquad \qquad \langle \text{do statement} \rangle$

# ANEXO B

TITTEL, Ed; GAITHER, Mark. **60 minutos para aprender Java**. São Paulo:Berkeley, 1996. Apêndice B.

## Apêndice Gramática Java

Esta é uma  
programas do Ja  
generalizar desd  
auxiliares quant

A gramática  
String e Charact  
indica a presenç

Cada regra t  
não-termina

Para esclarec  
linha e é separa

Outras met

- :| para alter
- (...) para a
- Sufixo ? pa
- Sufixo + pa
- Sufixo \* pa

Trata-se de  
regras de produçã  
pontuações precis  
específica para lin

```

Variable
  Identifier
;
Variable
  Expression
  | '('
;
Static
  Identifier
  'static'
;
Parameter
  Identifier
;
Parameter
  Identifier
  Type Identifier
;
Statement
  Variable
  Expression
  '(' Statement
  'if' '(' Statement
  'while'
  'do' Statement
  'for'
  'try' Statement
  'finally'
  'switch'
  '{'
  '}'
  'synchronized'
  'return'
  'throw'
  Identifier
  'break'
  'continue'
  ';'
;
Expression
  Expression
  Expression
  Expression
  Expression
  Expression
  Expression
  Expression
  Expression
  Expression
  Expression

```

```

ArgList
  Expre
;

Type =
  Types
;

TypeSpec
  'bool
  'byte
  'char
  'short
  'int'
  'float
  'long
  'double
  'class
  'inter
;

Modifier
  'public
  'private
  'protected
  'static
  'final'
  'native
  'synchronized
  'abstract
  'thread
  'transient
;

PackageName
  Identif
  Packag
;

ClassName
  Identif
  Packag
;

Interface
  Identif
  Packag
;

IntegerLiteral
  ('1'|'..'
  '0'('1'
  '0'('x'

```



## Gabarito das questões objetivas do ENADE e POSCOMP

Nº Alternativa correta

- 5 A
- 6 B
- 7 D
- 8 D (contra exemplo = 'b')
- 9 E
- 10 C
- 11 A
- 12 D
- 13 D
- 14 D
- 15 C
- 16 A
- 17 B
- 17 C
- 21 D
- 22 B
- 25 A
- 29 E
- 32 E
- 33 B (o correto seria VFVVF)
- 34 C
- 38 D
- 39 B
- 40 A
- 43 A
- 44 E
- 45 D
- 47 D
- 49 D

- 56 B
- 62 D
- 67 D
- 68 E
- 69 C
- 72 E
- 73 A
- 74 E
- 76 C
- 77 D
- 78 B
- 79 anulada
- 80 A