Compiladores – Prova #1

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**: Na análise linear/léxica, o fluxo de caracteres que compõem o programa fonte é lido, da esquerda para direita, e agrupado em tokens.

2. (5 pontos) Considere a gramática livre de contexto G abaixo:

$$T \to W \textbf{ sup } T \mid W \textbf{ sub } T \mid \epsilon$$
$$W \to x \mid y \mid z$$

(A) Quais são os não-terminais de G?

Solução: T e W

(B) Quais são os terminais de G?

Solução: sup, **sub**, x, y, z e ϵ

(C) Qual é o símbolo de partida de G?

Solução: T

3. (5 pontos) **Assinale a alternativa correta**. Em geral, quantas são as fases de um compilador que compõem a interface de vanguarda?

(A) 2

(B) 3

(X) 4

(C) 6

4. (5 pontos) **Assinale a alternativa correta**. Seja $\diamond$ um operador binário, associativo à direita, tal que $\alpha \diamond \omega = \alpha + 2\omega$. Qual é o valor da expressão $1 \diamond 2 \diamond 3$?

(A) 7

(B) 9

(C) 11

(X) 17

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

(F) Em uma árvore gramatical, os nós interiores são rotulados por um terminal.

(V) A gramática $S \to aS \mid Sa \mid a$ é ambígua.

(V) Um atributo de um nó de uma árvore gramatical é dito sintetizado se o seu valor depende apenas dos valores dos atributos dos seus nós filhos.

(V) Nos analisadores sintáticos $bottom\text{-}up$ a construção da árvore sintática parte das folhas em direção à raiz.

(F) Um analisador gramatical recursivo descendente é dito preditivo se a gramática a ser analisada não é recursiva à esquerda.

Parte B

6. (10 pontos) Na linguagem de programação Shell Script comando `until` possui a forma

```
until [ expression ]
do
    block
done
```

O comando `until` executa o bloco de código (**`block`**, na notação acima) enquanto a expressão (**`expression`**) é avaliada como falsa.

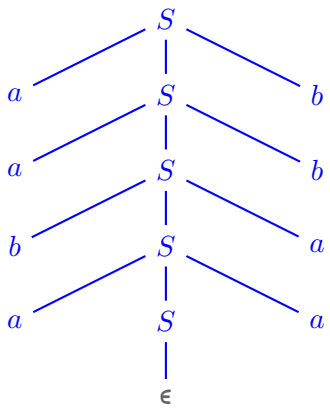
Construa um gabarito para a tradução dirigida pela sintaxa que traduz o comando `until` da linguagem Shell Script para código de máquina de pilha.

rótulo loop
código para expression
gotrue exit
código para block
goto loop
rótulo exit

7. (15 pontos) Considere a gramática G abaixo:

$$S \to aSb \mid bSa \mid aSa \mid bSb \mid \epsilon$$

Construa uma árvore gramática para a cadeia $aabaaabb$.



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

$$A \to Aa \mid Bb \mid c$$
$$B \to aA \mid bB \mid \epsilon$$

Reescreva a gramatical G, eliminando a recursão à esquerda.

Solução:

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

Parte C

9. (30 pontos) Considere a gramática G abaixo:

$$A \to aAb \mid abB \mid c$$
$$B \to bA \mid aB \mid \epsilon$$

Implemente em C, C++ ou Python, as funções associadas ao não-terminais de G para um analisador sintático recursivo descendente. 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 e deve ter, no máximo, 40 linhas. Para fins de implementação, assuma que os terminais a, b, c sejam representados, no código, pelos caracteres ASCII `'a'`, `'b'` e `'c'`, respectivamente.

Solução em C/C++:

```
1 void A()
2 {
3     switch (lookahead) {
4     case 'a':
5         reconhecer('a');
6
7         if (lookahead == 'b')
8         {
9             reconhecer('b');
10            B();
11        } else
12        {
13            A();
```

```
            reconhecer('b');
        }
        break;

    case 'c':
        reconhecer('c');
        break;

    default:
        erro();
    }
}

void B()
{
    switch (lookahead) {
    case 'a':
        reconhecer('a');
        B();
        break;

    case 'b':
        reconhecer('b');
        A();
    }
}
```

Solução em Python:

```
def A():

    if lookahead == 'a':
        reconhecer('a')

        if lookahead == 'b':
            reconhecer('b')
            B()
        else:
            A()
            reconhecer('b')
    elif lookahead == 'c':
        reconhecer('c')
    else:
        erro()


def B():

    if lookahead == 'a':
        reconhecer('a')
        B()
    elif lookahead == 'b':
        reconhecer('b')
        A()
```