Nome: **Prontuário:**

Data: / / **Grupo Experimental:**

Experimento 1 - Pré-Teste - Estrutura de Dados

Instruções do Experimento

- Identifique de qual grupo você faz parte (**GC, GE1 ou GE2**).
- O sorteio dos grupos foi realizado em sala de aula.
- **Questões teóricas**:
  - Todos os Grupos → não poderá consultar nenhuma fonte.
- **Questão prática**:
  - Todos os Grupos → poderá consultar apenas **material próprio**.
- Esta avaliação **não impactará diretamente nas notas finais** da disciplina; ela conta apenas como **nota de participação**.
- O descumprimento das regras implicará em **perda da nota de participação**.

Recursividade

1. Quais são as duas partes essenciais de uma função recursiva?

   Caso base e passo recursivo.

Arrays

2. Em C++, qual das alternativas descreve corretamente um array?

   a) Um array é uma coleção de variáveis que podem ter tipos diferentes.

   b) Um array armazena dados em posições de memória não adjacentes.

   c) Um array permite armazenar múltiplos valores do mesmo tipo em posições adjacentes de memória.

   d) Um array é um tipo de dado que permite o uso de nomes dinâmicos para acessar os elementos.

Listas

3. Qual a principal diferença entre uma lista encadeada simples e uma lista duplamente encadeada?

   A lista duplamente encadeada possui um ponteiro apontando para nó o anterior.

Pilha e Fila

4. Em uma estrutura de dados do tipo pilha (stack), a ordem de acesso aos elementos é:

   a) FIFO

   b) LIFO

   c) FILO

   d) LILO

Árvores

5. Considerando os conceitos de Árvores. Defina os termos:

   A. Folha
   Nó que não possui filhos.

   B. Filhos
   Nós diretamente conectados a um nó superior.

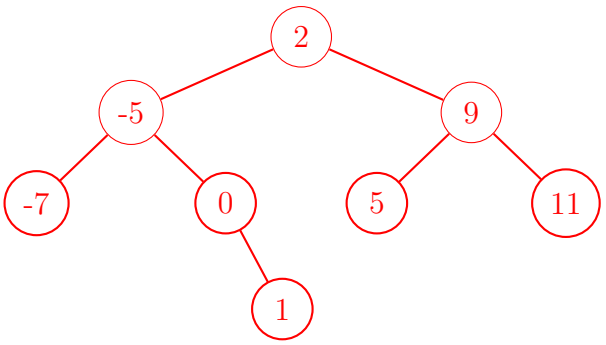
   C. Altura da árvore
   Número máximo de arestas da raiz até uma folha.

   D. Grau de um nó
   Número de filhos diretos que o nó possui.

Árvore Binária de Busca

6. Considere a seguinte sequência de inserções em uma árvore binária de busca vazia: 3, -3, 9, 11, -5, 5, 0, -7, 2, 1, remova os nós -3 e 3 caso necessário use o conceito de substituir o nó a ser retirado pelo nó mais à direita da subárvore da esquerda.

   A. Desenhe a árvore resultante após todas as inserções e remoções.



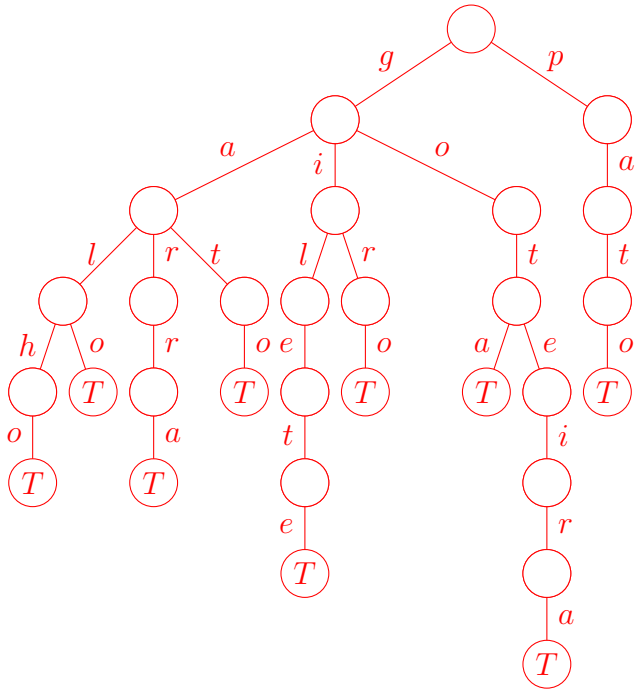
   B. Qual é o percurso pós-ordem da árvore final?
   -7, 1, 0, -5, 5, 11, 9, 2

   C. Qual é o percurso pré-ordem da árvore final?
   2, -5, -7, 0, 1, 9, 5, 11

   D. Qual é o percurso em largura (BFS) da árvore final?
   2, -5, 9, -7, 0, 5, 11, 1

Tries

7. Dado o conjunto de palavras { ”gato”, ”goteira”, ”galho”, ”gilete”, ”galo”, ”garra”, ”gota”, ”giro”, ”pato”}, desenhe uma Trie representando todas elas.



Árvore AVL

8. Preencha com (V) Verdadeiro ou (F) Falso:

   a) (F) Em uma árvore AVL, o fator de balanceamento de um nó pode ser 4, mas terá que ser balanceada.

   b) (V) Árvores AVL são sempre árvores binárias de busca.

   c) (F) A remoção de um nó nunca altera o balanceamento de uma árvore AVL.

   d) (V) Uma árvore AVL tem altura sempre limitada a O (log n).

Prática

9. Exercício 1 – Identificação de Rotações em uma Árvore AVL

   Implemente um programa que realize a seguinte tarefa:

   - Leia, de uma única vez, **cinco números inteiros**.
   - Insira esses números, **na ordem digitada**, em uma árvore AVL inicialmente vazia.
   - A cada inserção, verifique se houve necessidade de balanceamento da árvore.
   - Exiba, na ordem em que ocorrerem, as **rotações realizadas para manter a árvore balanceada**.

   Tipos de rotações a considerar:

   - **LL** – rotação simples à direita
   - **RR** – rotação simples à esquerda
   - **LR** – rotação dupla (esquerda-direita)
   - **RL** – rotação dupla (direita-esquerda)
   - **Nenhuma** – caso a árvore já esteja balanceada

   Exemplo de saída esperada:

RL, LL, RL, LR.

```
// Primeiro peguei o código base de árvore AVL, disponível no classroom.
// Depois adaptei para fazer o que o exercício pede.

#include <iostream>
using namespace std;

// Variável global para controlar se já foi exibida uma rotação.
bool rodou = false;

struct NoAVL
{
    int chave;
    int fb;
    NoAVL* dir;
    NoAVL* esq;
};

NoAVL* CriarNo(int chave){
    NoAVL* novoNo = new NoAVL;
    novoNo->chave = chave;
    novoNo->dir = nullptr;
    novoNo->esq = nullptr;
    novoNo->fb = 0;
    return novoNo;
}

NoAVL* RotacaoL(NoAVL* p){
    NoAVL* u = p->esq;

    if (u->fb == -1)
    {
        /* Rotação LR */
        NoAVL* v = u->dir;
        u->dir = v->esq;
        v->esq = u;
        p->esq = v->dir;
        v->dir = p;

        if (v->fb == 1) {
            u->fb = -1;
            p->fb = 0;
        } else if (v->fb == -1) {
            u->fb = 0;
            p->fb = 1;
        } else {
            u->fb = 0;
            p->fb = 0;
        }
```

```
        v->fb = 0;

        // Exibe o tipo de rotação realizada
        // (Forma para ficar com ponto final correto).
        if (rodou)
            cout << ", LR";
        else
            cout << "LR";

        // Exibe o tipo de rotação realizada
        // (Forma sem se preocupar com o ponto).
        //cout << "LR, ";

        // Marca que já foi exibida uma rotação.
        rodou = true;
        return v;
    }

    /* Rotação LL */
    p->esq = u->dir;
    u->dir = p;
    p->fb = 0;
    u->fb = 0;

    // Exibe o tipo de rotação realizada
    // (Forma para ficar com ponto final correto).
    if (rodou)
        cout << ", LL";
    else
        cout << "LL";

    // Exibe o tipo de rotação realizada
    // (Forma sem se preocupar com o ponto).
    //cout << "LL, ";

    // Marca que já foi exibida uma rotação.
    rodou = true;
    return u;
}

NoAVL* RotacaoR(NoAVL* p){
    NoAVL* u = p->dir;

    if (u->fb == 1)
    {
        /* Rotação RL */
        NoAVL* v = u->esq;
        u->esq = v->dir;
        v->dir = u;
        p->dir = v->esq;
```

```
        v->esq = p;

        if (v->fb == 1) {
            p->fb = -1;
            u->fb = 0;
        } else if (v->fb == -1) {
            p->fb = 0;
            u->fb = 1;
        } else {
            p->fb = 0;
            u->fb = 0;
        }

        v->fb = 0;

        // Exibe o tipo de rotação realizada
        // (Forma para ficar com ponto final correto).
        if (rodou)
            cout << ", RL";
        else
            cout << "RL";

        // Exibe o tipo de rotação realizada
        // (Forma sem se preocupar com o ponto).
        //cout << "RL, ";

        // Marca que já foi exibida uma rotação.
        rodou = true;
        return v;
    }

    /* Rotação RR */
    p->dir = u->esq;
    u->esq = p;
    p->fb = 0;
    u->fb = 0;

    // Exibe o tipo de rotação realizada
    // (Forma para ficar com ponto final correto).
    if (rodou)
        cout << ", RR";
    else
        cout << "RR";

    // Exibe o tipo de rotação realizada
    // (Forma sem se preocupar com o ponto).
    //cout << "RR, ";

    // Marca que já foi exibida uma rotação.
    rodou = true;
```

```
    return u;
}

NoAVL* Inserir (NoAVL*& raiz, int chave, bool& cresceu){

    if(raiz == nullptr){
        raiz = CriarNo(chave);
        cresceu = true;
        return raiz;
    }

    else if(chave < raiz->chave){
        Inserir (raiz->esq, chave, cresceu);
        if (cresceu){
            if (raiz->fb == 0)
                raiz->fb = 1;
            else if (raiz->fb == -1)
                raiz->fb = 0;
            else if (raiz->fb == 1) {
                raiz = RotacaoL (raiz);
                cresceu = false;
            }
        }
    }
    else {
        Inserir (raiz->dir, chave, cresceu);
        if (cresceu){
            if (raiz->fb == 0)
                raiz->fb = -1;
            else if (raiz->fb == 1)
                raiz->fb = 0;
            else if (raiz->fb == -1){
                raiz = RotacaoR (raiz);
                cresceu = false;
            }
        }
    }

    return raiz;
}

int main()
{
    NoAVL* raiz = nullptr;
    bool cresceu = false;

    // Leia, de uma única vez, cinco números inteiros.
    int numeros[5];

    cout << "Digite cinco números inteiros: ";
```

```
    for (int i = 0; i < 5; i++)
        cin >> numeros[i];


    // Insira esses números, na ordem digitada,
    // em uma árvore AVL inicialmente vazia.
    for (int i = 0; i < 5; i++)
        Inserir (raiz, numeros[i], cresceu);

    if (!rodou)
        cout << "Nenhuma"; // Nenhuma rotação ao inserir os números.

    cout << "." << endl; // Ponto final da frase.

    return 0;
}
```