Disciplina: **Estrutura de dados 2**
Professor: **Jeferson José Baqueta**

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Atividade prática 1

Especificações:

- Esse trabalho é individual;
- O valor total do trabalho representará 10% da nota do bimestre;
- O trabalho deve ser entregue em formato digital no Moodle (data da entrega: 22/09):
  - Arquivos .c compactados.
- Não serão aceitos, em hipótese alguma, trabalhos enviados via e-mail.
- Cópias de trabalhos serão penalizadas com a perda total do valor do trabalho.
- O trabalho enviado deve estar claramente identificado, com nome do aluno.

Objetivo:

Implementar as operações de inserção, busca e remoção em uma Árvore de Busca Binária (BST) utilizando a linguagem de programação C.

Estrutura da árvore

A árvore BST será composta por nodos. Cada nodo possui um valor inteiro (chave), dois ponteiros, um para o nodo da esquerda e outro para o nodo da direita e um ponteiro para o nodo pai. A estrutura de um nodo é apresentada abaixo:

```
typedef struct No{
    int chave;
    struct No* pai;
    struct No* direita;
    struct No* esquerda;
} Nodo;
```

Inserção

Implementar uma função **inserir**, a qual recebe um valor inteiro como argumento (chave) e insere essa chave na árvore BST. Certifique-se de manter as propriedades da árvore BST após a inserção. Se o valor já existir na árvore, a função deve lidar com isso de maneira apropriada (alocar a chave replicada em uma das subárvores da raiz).

Busca

Implementar uma função **buscar**, a qual recebe um valor inteiro como argumento (chave de busca) e retorna verdadeiro se a chave estiver presente na árvore BST. Se a chave não estiver na árvore, sua função deve retornar falso.

Remoção

Implementar uma função **remover**, a qual recebe um valor inteiro como argumento (chave) e remove o nodo correspondente da árvore BST. Certifique-se de manter as propriedades da árvore BST após a

remoção. Se a chave não estiver na árvore, a função deve retornar NULL. A escolha do nodo substituto para remoção fica ao critério do aluno (antecessor ou sucessor).

Entrada e saída:

Seu programa deve fazer a leitura de um arquivo de texto com o seguinte formato:

i 23

i 20

i 25

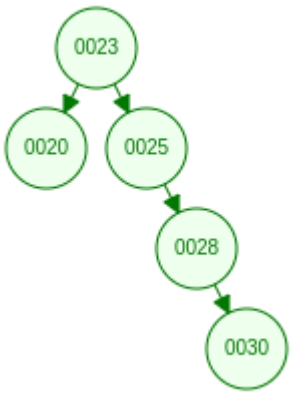
i 28

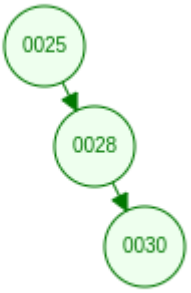
i 30

r 20

r 23

Cada linha desse arquivo representa uma operação de inserção (i) ou remoção (r). A operação de busca será avaliada em conjunto com a remoção, uma vez que a operação de remoção é precedida por uma operação de busca. Você deverá montar sua árvore BST realizando as operações descritas no arquivo de texto (executando as operações em sequência). O nome do arquivo de texto de entrada deve ser "in.txt". Por exemplo, se realizarmos as operações descritas acima em sequência, a árvore resultante será a seguinte:





i 23, i 20, i 25, i 28, i 30 r 20, r23

Como saída seu programa deve imprimir a árvore resultante (**em ordem**) em um arquivo chamado "out.txt", como apresentado abaixo:

25

28

30

Documentação e Comentários:

Documente seu código de maneira clara, incluindo comentários explicativos em todas as funções. Certifique-se de que seu código siga boas práticas de programação, como nomes de variáveis significativos e formatação adequada.