Árvores Binárias de Pesquisa

Considere as seguintes definições dos tipos de dados **INFOABP**, **NodoABP** e **PNodoABP**:

```
typedef struct {
    int  numAluno;        // número de aluno (chave): { 70000, …, 75000 }
    float  notasMTP[2];   // notas dos mini-testes práticos: [0.0, 2.0]
    float  notasTE[2];    // notas dos testes escritos: [0.0, 8.0]
    int  notaFinal;       // nota final (valor arredondado da soma das 4 notas): { 0, …, 20 }
} INFOABP;

struct NodoABP {
    INFOABP  Elemento;
    struct NodoABP  *Esquerda;
    struct NodoABP  *Direita;
};
typedef struct NodoABP *PNodoABP;
```

Cada elemento (registo) do tipo **INFOABP** corresponde aos dados sobre a avaliação de um aluno à disciplina de AED, durante a época de Aprendizagem. O campo **numAluno** é a chave (único).

Descarregar as seguintes bibliotecas (**Folhas práticas ---> Bibliotecas e Exercícios das folhas práticas ---> Árvores Binárias de Pesquisa**):

Aleatorio.h ---> com as operações para gerar números aleatoriamente

OperacoesBasicasExABP.h ---> com a estrutura **INFOABP** e as operações básicas sobre ela

EADArvoreBinariaPesquisa.h ---> com os tipos e as operações básicas sobre **ABP's**

OperacoesExABP.h ---> com as operações do exercício desta folha prática

MainExABP.c ---> com o programa principal (main) dos exercícios desta folha prática

Mostrar uma ABP por níveis (blibliotecas excusivas desta operação):

ABPPorNiveis.h ---> com a operação **mostrarPorNiveisABP**

EADFila.h ---> com as operações sobre a EAD Fila

Elaborar um programa em C que utilize as operações contidas nas bibliotecas referidas e resolva as questões colocadas a seguir, acrescentado-as uma a uma ao programa principal (main).

1. Implementar uma função que

- receba dois valores inteiros positivos, **A** e **B**,
- devolva uma ABP **T** com elementos do tipo **INFOABP**, gerados aleatoriamente, cuja quantidade é um número a variar entre **A** e **B** (usar operações da biblioteca **Aleatorio**).

Usar esta função para construir uma ABP com elementos o tipo **INFOABP**, cuja quantidade de elementos é um valor entre 0 e 15. Mostre a ABP T por níveis e desenhe a árvore T no papel.

2. Acrescente ações ao programa que
- mostre a raiz de **T** e
- mostre todos os elementos de **T** usando travessia em-ordem

3. Acrescente ações ao programa que
- determine e mostre a **altura** e a **quantidade de níveis** da ABP T

4. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- devolva o elemento de **T** com maior valor da chave (campo **numAluno**).

5. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- devolva o elemento de **T** com menor valor da chave (campo **numAluno**).

6. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP** e um valor inteiro **A**,
- devolva o número de elementos de **T** cujos valores do campo **numAluno** são superiores a **A**.

7. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP** e dois inteiros **A** e **N**,
- devolva o número de elementos de **T** com valores do campo **numAluno** são superiores a **A** e com valores no campo notaFinal menor ou igual a N.

8. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP** e um valor inteiro **A**,
- devolva o elemento que contém o elemento com valor do campo **numAluno** igual a **A**, caso exista; se não existir, devolver um elemento com valor negativo no campo **numAluno**

9. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP** e dois valores inteiros **A** e **B** (A < B),
- devolva o número de elementos de **T** cujos valores do campo **numAluno** são maiores ou iguais a **A** e menores ou iguais a **B**.

10. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP** e dois valores inteiros **A** e **B** (A < B),
- devolva o número de elementos de **T** cujos valores do campo **numAluno** são menores que A ou maiores que B.

11. Acrescentar ao programa ações que
- remova da ABP **T** um qualquer elemento **X** do tipo **INFOABP**, caso exista
- mostre a árvore obtida após aquela operação.

12. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- transforme esta ABP num array 1D (vetor) ordenado crescentemente.

13. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- devolva o nível da folha mais próxima (em altura) da raiz de **T**.

Ao testar esta função no programa principal, determine também o nível da folha mais afastada (em altura) da raiz da ABP.

14. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- remova o maior elemento (considerando valor no campo **numAluno**) de **T** (otimizar o algoritmo, percorrendo a ABP uma unica vez).

15. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- remova o menor elemento (considerando valor no campo **numAluno**) de **T** (otimizar o algoritmo, percorrendo a ABP uma unica vez).

16. Implementar uma função que
- receba uma ABP **T** com elementos do tipo **INFOABP**,
- insira um elemento **X** do tipo INFOABP (gerado aleatoriamente) na ABP **T**.