Árvores Binárias

Considere as seguintes definições de tipos de dados **DadosAB**, **NodoAB** e **PNodoAB** (aulas):

```
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 }
} DadosAB;

struct NodoAB {
    DadosAB  Elemento;
    struct NodoAB  *Esquerda;
    struct NodoAB  *Direita;
};
typedef struct NodoAB *PNodoAB;
```

Cada elemento (registo) do tipo **DadosAB** corresponde aos dados sobre a avaliação de um aluno à disciplina de AED, durante a época de Aprendizagem.

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

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

OperacoesBasicasAB.h ---> operações básicas sobre o tipo de dados **DadosAB**

EADArvoreBinaria.h ---> operações básicas sobre a **EAD Árvores Binárias**

OutrasOperacoesAB.h ---> operações a implementar para os exercícios propostos

MainAB.c ---> **programa principal** relativo aos exercícios propostos

Mostrar uma AB por níveis (bibliotecas exclusivas desta operação):

ABPorNiveis.h ---> com a operação **mostrarPorNiveisAB** (com definição do tipo DadosFila)

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 AB **T** com elementos do tipo **DadosAB**, gerados aleatoriamente, cuja quantidade é um número a variar entre **A** e **B** (usar operações da biblioteca **Aleatorio**).

Usar esta função para construir uma AB com elementos o tipo **DadosAB**, cuja quantidade de elementos é um valor entre 0 e 15.

2. Mostre a AB construída em 1, usando as 3 travessias estudadas: em-ordem, pré-ordem e pós-ordem. Mostre também esta AB por níveis.

3. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva a **quantidade** de elementos (nodos) de **T**.

4. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB** e um valor inteiro **K**,
- **-** devolva a **quantidade** de elementos de T com valor no campo **notaFinal** igual a **K**.

5. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva a **soma** dos valores do campo **notaFinal** de todos os elementos de **T**.

Usar esta função para determinar a **média** dos valores do campo **notaFinal** dos elementos de uma AB (programa principal).

6. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva o **maior valor** no campo **notaFinal** dos elementos de **T**.

7. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva o **elemento** de **T** com **menor valor** no campo **numAluno**.

8. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB,**
- **-** construa e devolva uma nova AB do mesmo tipo que seja uma **cópia** de T.

9. Implementar uma função que
- **-** receba duas AB, **T1** e **T2,** com elementos do tipo **DadosAB**,
- **-** devolva o valor **1** se são **iguais** (T1 = T2) e **0** caso contrário.

10. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva a **quantidade** de nodos de T sem filhos (que são **folhas**).

11. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB**,
- **-** devolva a **quantidade** de nodos de T com **um e um só filho**.

12. Implementar uma função que
- **-** receba uma AB **T** com elementos do tipo **DadosAB** e um elemento **X** do tipo **DadosAB**,
- **-** devolva o **nível** a que se encontra do elemento **X** em **T**.

13. Implementar uma função que
- **-** receba uma árvore binária **T** com elementos do tipo **DadosAB,**
- **-** construa e devolva uma nova **AB** do mesmo tipo que seja a imagem **espelhada** de T (isto é, todas as subárvores da esquerda são agora subárvores da direita e vice-versa).