

1- Dada a árvore do diagrama acima, indique...

• os nós folha.

**D, J, E, T**

• o grau da árvore.

**O grau é 2, que são H, R**

• a altura da árvore.

**Altura da arvore é 3**

• os descendentes do nó H.

**São D e J**



2- ENADE 2005 questão 39b. Descreva a ordem de visita para uma varredura em pré-ordem (r-e-d, ou pré-fixada à esquerda) na árvore acima.

**Primeiro insere em 33 que a raiz, 15 a esquerda porque é menor que 33 e menor que 41.**

**41 fica à direita porque é maior que o 33 e 15.**

**38 fica à esquerda e abaixo do 41 porque é menor que 41 e menor que 47.**

**34 fica à esquerda e abaixo 38 porque é menor 38 e 47.**

**47 fica abaixo e a direita do 41 porque é maior que 41, 38 e 34.**

**O 43 fica abaixo e esquerda do 47 porque é menor 47 e 49.**

**49 fica abaixo e direita do 47 porque é maior que 47 e 43.**

**Sequência final: 33-15-41-38-34-47-43-49.**

3- Duas árvores binárias (ABs) são SIMILARES se possuem a mesma distribuição de nós (independente dos valores nos mesmos). Em uma definição mais formal, duas ABs são SIMILARES se são ambas vazias, ou se são ambas não vazias e suas sub-árvores esquerdas são similares, e suas sub-árvores direitas também são similares. Implemente a função: SIMILARES (por referência A1, A2: do tipo AB): resultado do tipo boolean.

**função: SIMILARES (por referência A1, A2: do tipo AB): resultado do tipo boolean.**

**se A e B = NULL**

**então**

**retorna verdadeiro**

**senão**

**se (A = NULL e B <> NULL) ou (A <> NULL e B = NULL)**

**então**

**retorna falso**

**senão**

**se SIMILARES(direita (A), direita (B)) e SIMILARES (esquerda (A),esquerda (B))**

**então**

**retorna verdadeiro**

**senão**

**retorna falso**

**fim\_se**

**fim\_se**

**fim\_se**

**Fim função**

4- Duas ABs são IGUAIS se são ambas vazias, ou se armazenam valores iguais em suas raízes, e suas sub-árvores esquerdas são iguais, e suas sub-árvores direitas são iguais. Implemente a função: IGUAIS (por referência A1, A2: do tipo AB): resultado do tipo boolean.

**IGUAIS(por referencia A,B : tipo AB): booleano**

**se A e B = NULL**

**então**

**retorna verdadeiro**

**senão**

**se (A = NULL e B <> NULL) ou (A <> NULL e B = NULL)**

**então**

**retorna falso**

**senão**

**se (info(a)=info(b) e iguais(direita (a),direita (b)) e iguais(esquerda(a),esquerda (b))**

**então**

**retorna verdadeiro**

**senão**

**retorna falso**

**fim\_se**

**fim\_se**

**fim\_se**

**Fim-Função.**

5- Uma AB é estritamente binária se todos os nós da árvore tem 0 filhos ou 2 filhos. Implemente uma função que veri­fica se uma ABB é estritamente binária: ESTRITAMENTE\_BINÁRIA (por referência A1: do tipo AB): resultado do tipo boolean.

**Resposta:**

**ESTRITAMENTE\_BINARIA(por referencia A1 : tipo AB): boolean**

**se A1= NULL**

**então**

**retorna verdadeiro**

**senão**

**se (direita(a)= NULL e esquerda(a)<> NULL) ou (direita(a) <> NULL e esquerda(b)= NULL)**

**entao**

**retorna falso**

**senao**

**retorna ( ESTRITAMENTE\_BINARIA(direita(a)) e ESTRITAMENTE\_BINARIA(esquerda(a)))**

**Fim-se**

**Fim-se**

**Fim Função.**

6- Implemente uma função para testar se uma árvore binária AB é uma ABB. É\_ABB? (Por referência A: do tipo AB): resultado do tipo boolean.

**Resposta:**

**Implemente uma função para testar se uma árvore binária é uma ABB.**

**Definição: Árvore Binária de Busca - ABB**

**Uma árvore binária com raiz R é uma ABB se:**

**(1) A chave (informação) de cada nó da sub-árvore esquerda de R é menor (\*) do que a**

**chave (informação) do nó R;**

**(2) A chave de cada nó da sub-árvore direita de R é maior (\*) do que a chave do nó R;**

**(3) As sub-árvores esquerda e direita também são ABBs.**

**Eh\_ABB? (por referência A : do tipo árvore\_binária ) : resultado do tipo boolean.**

**se A = NULL //**

**então**

**retorne verdadeiro**

**senão se (TemAlguemMaior( Esq(A), Info(A) ) OU TemAlguemMenor( Dir(A), Info(A) )**

**então**

**retorne falso**

**senão**

**retorne ( É\_ABB( Esq(A) ) AND É\_ABB( Dir(A) ) )**

**Fim-se**

**Fim-se**

**Fim-se**

**Fim Função**

**Bool TemAlguemMaior( AB R, int valor )**

**se R = NULL**

**então**

**retorne falso**

**senão se Info( R ) > Valor**

**então**

**retorne verdadeiro**

**senão**

**retorne (TemAlguemMaior( Esq( R ), Valor ) OU TemAlguemMaior( Dir( R ),**

**Valor ) )**

**Fim-se**

**Fim-se**

**Fim-se**

**Fim Função**

7- Implemente uma função para determinar a altura de uma AB.

Altura (por referência A: do tipo AB): resultado do tipo Inteiro.

**ProfunDireita, ProfunEsquerda : inteiros // guarda a profundidade dos ramos dir e esq.**

**se A = null**

**então**

**retorna 0**

**senão**

**ProfunDireita = 1+altura(direita (a))**

**ProfunEsquerda = 1+altura(esquerda(a))**

**se ProfunDireita > ProfunEsquerda**

**entao**

**retorna ProfunDireita**

**senao**

**retorna ProfunEsquerda**

**fim\_se**

**fim\_se**

**Fim função**

8- Implemente uma função para determinar a soma dos valores de todos os nós de uma árvore binária com valores inteiros.

**Inteiro\_Soma(por referencia A: do tipo AB)**

Variavel temporária TotalSoma: tipo inteiro

Se (A <> NULL) então

TotalSoma = INFO(A)+Inteiro\_Soma(ESQ(A))+Inteiro\_Soma(DIR(A))

Retorne TotalSoma

Senão

Retorne 0

fim-se

**Fim-função**

9- Implemente uma função para determinar quantos nós de uma árvore binária com valores inteiros são maiores que zero.

**Inteiro QuantosMaioresQueZero (por referência A : do tipo AB )**

Variavel temporária QtdMZero: tipo inteiro

Se (A <> NULL) então

Se (INFO(A)>0) ENTÃO

QtdMZero = 1 + **QuantosMaioresQueZero(**ESQ(A)**) + QuantosMaioresQueZero(**DIR(A)**)**

Senão

QtdMZero = 0 + **QuantosMaioresQueZero(**ESQ(A)**) + QuantosMaioresQueZero(**DIR(A)**)**

fim-se

Retorne QtdMZero

Senão

Retorne 0

fim-se

**Fim-função**

10- Implemente uma função para contar quantos nós de uma árvore binária têm um único filho.

**Inteiro QuantosNósCom1ÚnicoFilho (por referência A : do tipo AB )**

Se (A = NULL) OU (ESQ(A) = NULL) E (DIR(A)=NULL) então

Retorne 0

Senão

Retorne 1 + **QuantosNósCom1ÚnicoFilho(**ESQ(A)**) + QuantosNósCom1ÚnicoFilho(**DIR(A)**)**

fim-se

**Fim-função**