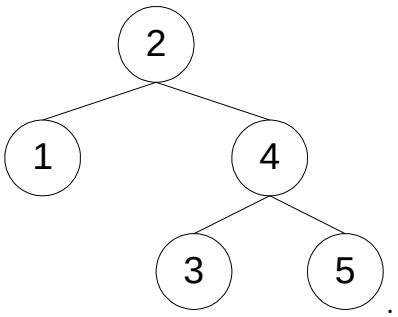
Árvores Binárias de Pesquisa

1. Considere as classes treeNode, bst e as classes iterator segundo as diferentes travessias numa árvore.
   - **a)** Defina uma classe tree por herança da classe bst.
   - **b)** Usando os métodos das várias classes iterator adicione à classe tree os métodos de visita: preOrder(), inOrder(), posOrder() e byLevel().
   - **c)** Crie a seguinte árvore:



   e teste os métodos de travessia.

   Adicione à classe tree os seguintes métodos:
   - **a)** Contar os nós de cada nível da árvore
   - **b)** Sobrecarga do operador atribuição
   - **c)** Imprimir os n maiores elementos da árvore

2. Faça a ordenação de uma lista usando uma árvore binária de pesquisa. Teste o método com uma lista inicializada com 10 elementos inteiros gerados aleatoriamente.

Prática - Laboratorial 5

Árvores Binárias de Pesquisa , Árvores AVL, Heaps

Estruturas de Informação

Árvores AVL

1. Teste a classe avlst:
   a) Construa uma árvore com os elementos: A, Z, B, C, X, D, P, E, V, F, Q .
   b) Remova os elementos: B, A, F, P, C.

2. Implemente um método externo à classe avlst<TN> que permita aceder ao penúltimo maior elemento da árvore.

3. Implemente um método na classe avlst<TN> que indica se a árvore AVL é uma árvore perfeitamente equilibrada, ou seja, uma árvore em que todos os nós têm feq=0.

Heaps

1. Verifique se o seguinte vetor corresponde a uma heap; caso contrário, construa a heap, com os elementos:

   20, 10, 12, 2, 8, 18, 9, 3, 5, 6

2. Adicione à classe template Heap<T> a sobrecarga do operador + que junta duas heaps.

3. Implemente uma função que faça a ordenação de um vetor usando a classe heap (algoritmo heapsort).