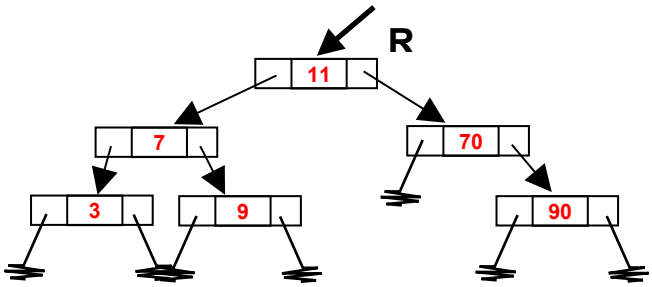
# Prova 2

Atividade Avaliativa (também usada para Cômputo de Frequência)
Algoritmos e Estruturas de Dados 1 (1001502)

**Orientações Gerais**
- **Tempo para elaboração da simulação / prova: 3h**.

**Orientações Quanto a Notação, Nomes das Variáveis, e Estruturas**
- Use os **mesmos nomes fornecidos no enunciado** (R, Ch, V, N, LastPosition, etc.). Utilize variáveis auxiliares temporárias, o tanto quanto forem necessárias. É só declarar e usar. Mas **não considere a existência de nenhuma outra variável permanente**, além das definidas no enunciado. Não considere prontas para uso nenhuma operação, salvo se explicitamente indicado no enunciado da questão.
- Considere as **estruturas exatamente conforme definido no enunciado**, seja no texto da questão, seja nos diagramas.
- Para o desenvolvimento de algoritmos, é possível usar a notação conceitual adotada no Livro Texto (p = NewNode; Deletenode(P), P->Info e P->Dir (filho direito de P), P->Esq (filho esquerdo de P), sendo P uma variável do tipo NodePtr - ponteiro para nó). Também é possível implementar em C ou C++.

**Questão 1 (2,5 pontos)** Considere uma ***Árvore Binária de Busca*** (ABB), de raiz R, implementada com alocação encadeada e dinâmica de memória, conforme os diagramas abaixo. A Árvore não contém elementos repetidos.





**árvore vazia**

Considere ainda que o nó de a árvore binária de busca tenha a seguinte declaração:

```
typedef struct node {
        int chave;
        struct node *esq;
        struct node *dir;
} Node;

typedef Node *  ABB;
```

Implemente a operação:

```
        void insere(ABB * R, int Ch);
        /* esta função deve inserir o elemento Ch na ABB R, caso o elemento já não estiver na árvore. */
```

**Questão 2 (2,5 pontos)** Considere uma ***Árvore Binária de Busca*** (ABB), de raiz R, implementada conforme os diagramas e declarações indicados na questão 1.

a) Escreva uma função que calcule a altura1 de uma árvore de raiz R. Para a responder esta prova, considere que a altura1 de **uma árvore vazia é zero**, e que a altura1 da árvore do diagrama a esquerda na Questão 1 é 3.

```
        int getAltura1(Node * R);
        //retorna a altura1 da árvore de raiz R.
```

b) Qual é a ordem de eficiência de tempo dessa função, em termos de número de comparações?

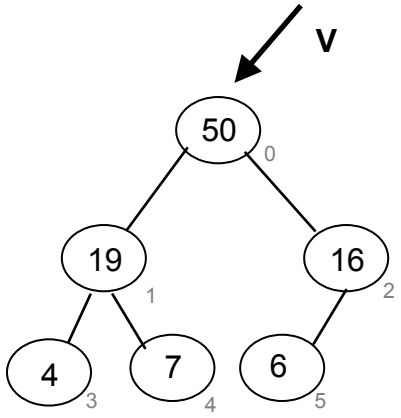
**Questão 3 (2,5 pontos)**

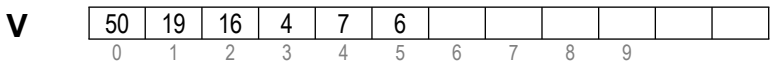
a) Implemente a função abaixo, que ordena um vetor V de inteiros, de tamanho N, colocando os elementos em ordem crescente. Use para isso um dos algoritmos estudados na disciplina – ordenação por Inserção, Seleção ou Bolha.

```
        void ordena(int * V, int N);
        /* ordena o vetor V de tamanho N */
```

b) Qual é nome do algoritmo que você implementou? Qual é a ordem de eficiência de tempo do mesmo, em termos de número de comparações?

**Questão 4 (2,5 pontos)** Em um ***Heap-Binário-de-Máximo***, o elemento que está em um determinado ***nó*** da árvore tem valor maior ou igual do que o valor de seus ***filhos*** direito e esquerdo. Entre os nós ***irmãos***, não há necessariamente uma ordenação, como mostra o diagrama, a seguir.




Escreva uma função que verifica se um vetor V[0 .. LastPosition] é um heap-binário-de-máximo, ou não. A função recebe como parâmetro o inteiro LastPosition, que indica a última posição do vetor que efetivamente contém elementos.

```
bool IsHeap(int * V, int LastPosition);
/* Verifica se o vetor V é um um heap-binário-de-máximo, retornando true caso sim, e false caso não.  */
```

Use as definições abaixo em sua função:

```
#define pai(i) ((i – 1) / 2 )
#define fesq(i) (i * 2  +  1)
#define fdir(i) (i * 2  +  2 )
```