1. Binary trees. Each node has 2 children which can be empty. It has also a parent which also can be empty.

A node without any children (both empty) is leaf. Other nodes are internal. The node without any parent is root.

A Pascal type representation:

```pascal
type node = record
  key : integer
  l, r : node
end

tree = node
```

2. More terminology. X is a descendant of U if it is a child of U or a descendant of a child of U.

If X is descendant of U, then U is an ancestor of X.

Depth of U is # edges from root to U. Height of U is max. # edges from U to a leaf. Subtree of U consists of U and its descendants.
3. Traversals. Three traversals are in-order, pre-order and post-order.

Procedure Inorder (M)
    if M ≠ nil then
        Inorder (M.l);
        print (M.key);
        Inorder (M.r);
    endif

Procedure Preorder (M)
    if M ≠ nil then
        print (M.key);
        preorder (M.l);
        preorder (M.r);
    endif

Postorder: first recurse left, then right, then print
property: \( \mu \) is ancestor of \( V \) iff
\[
\text{pre}(\mu) < \text{pre}(V) \quad \text{and} \quad \text{post}(\mu) > \text{post}(V)
\]

A binary tree is a binary search tree if the keys printed in order is sorted.
4. Searching.

function Search(x, m)
    if m = nil or x = m.key then return m
    else
        if x < m.key then return Search(x, m.l)
        else return Search(x, m.r)
    endif
endif

Write Successor and Min functions
5. Insertion and Deletion

```plaintext
procedure Insert (P, Y)
    x := nil; M := P;
    while M ≠ nil do x := x;
        if Y.key < M.key then M := M.l
        else M := M.r
    endwhile
    Y.p := x;
    if x = nil then P := Y
    else if Y.key < x.key then
        x.l := Y; else x.r := Y
    endif
endif
```

Deletion is slightly complicated:

Case 1: node has no children
Case 2: node has one child
Case 3: node has two children
a. replace key by successor key
b. delete successor