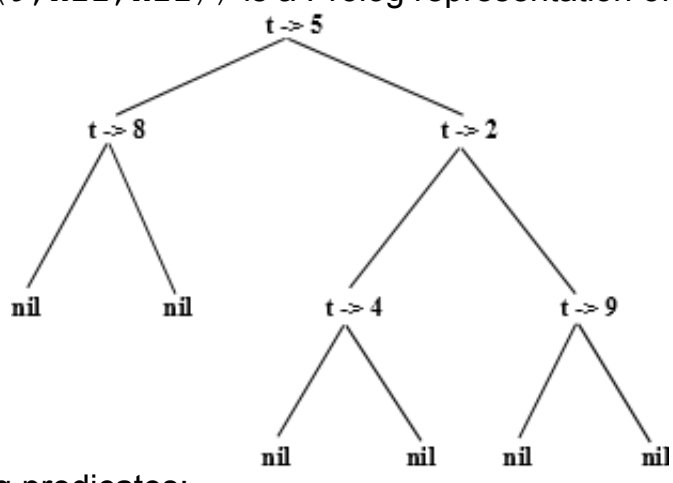
Exercise 1

In this exercise we want to process binary trees. A tree is represented by a compound term `t(Value,Left,Right)` where `Value` is the value of the root of `t`, `Left` is the left subtree of `t`, and `Right` is the right subtree of `t`. For example the compound term `t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))` is a Prolog representation of the following tree:



Write the Prolog program specifying the following predicates:

- **`is_a_tree/1`**. is_a_tree(T) is true if T is a tree of the form t(V,L,R) or if T is an empty tree denoted. For example:

```
?- is_a_tree(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil)))).
true.

?- is_a_tree(t(5,t(8,99,nil),t(2,t(4,nil,nil),t(9,nil,nil)))).
false.

?- is_a_tree(t(5,t(8,nil,nil),t(2,nil,nil),t(9,nil,nil))).
false.

?- is_a_tree(nil).
true.
```

- **`count_leaves/2`**. count_leaves(T,N) is true if N is the number of leaves of the tree T. For example:

```
?- count_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),N).
N = 3

?- count_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),3).
true.

?- count_leaves(t(5,t(8,nil,t(12,nil,nil)),t(2,t(4,t(27,nil,nil),t(30,nil,nil)),t(9,nil,nil))),N).
N = 4

?- count_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),6).
false.
```

- **`collect_leaves/2`**. collect_leaves(T,L) is true if L is the list containing all the leaves of the tree T. For example:

```
?- collect_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),L).
L = [8,4,9]

?- collect_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[8,4,9]).
true.

?- collect_leaves(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[8,4,9,12]).
false.
```

- **`collect_internal_nodes/2`**. collect_internal_nodes(T,L) is true if L is the list containing all the internal nodes of the tree T. For example:

```
?- collect_internal_nodes(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),L).
L = [5,2]

?- collect_internal_nodes(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[5,2]).
true.

?- collect_internal_nodes(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[5,2,78]).
false.
```

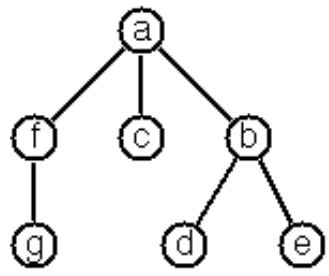
- **`collecte_nodes_level/3`**. collecte_nodes_level(T,L,N) is true if L is the list containing all the nodes (internal or leaves) of the tree T at level N. For example:

```
?- collecte_nodes_level(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),L,2).
L = [8,2]

?- collecte_nodes_level(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[8,2],2).
true.

?- collecte_nodes_level(t(5,t(8,nil,nil),t(2,t(4,nil,nil),t(9,nil,nil))),[8,99],2).
false.
```

Exercise 2

We now consider n-ary trees. A tree is represented by the compound term `t(Value,List)` where `Value` is the root value of the tree `t` and `List` is the list of all the subtrees of `t`. For example, the compound term `t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,`

`[ ] ) ] ) ] )` is a Prolog representation of the following tree:



Write the Prolog program specifying the following predicates:

- **`is_a_tree/1`**. is_a_tree(T) is true if T is a tree of the form t(V,L) or if T is an empty tree denoted. For example:

```
?- is_a_tree(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])])).
true.

?- is_a_tree(t(a,[t(f,[t(g,h)]),t(c,[]),t(b,[t(d,[]),t(e,[])])])).
false.
```

- **`count_nodes/2`**. count_nodes(T,N) is true if T is a tree that has N nodes. For example:

```
?- count_nodes(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])]),N).
N = 7

?- count_nodes(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])]),7).
true.

?- count_nodes(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])]),8).
false.
```

- **`length_internal_path/2`**. length_internal_path(T,N) is true if in the tree T the sum of the lengths of the paths to each of the nodes is equal to N. For example:

```
?- length_internal_path(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])]),L).
L = 9
```

- **`bottomup/2`**. bottomup(T,L) is true if L is the list of the nodes of the tree T traverse from bottom to top. For example:

```
?- bottomup(t(a,[t(f,[t(g,[])]),t(c,[]),t(b,[t(d,[]),t(e,[])])]),L).
L = [g, f, c, d, e, b, a].
```