AIT Semantic and Declarative Technologies Course
Class practice: Prolog data structures

In the problem set below we use the data structure *itree* (integer tree) which is defined as follows.
A Prolog term is an *itree* if and only if:

- it is a leaf, i.e., a `leaf(`n`)` compound, where n is an integer; or
- it is a node, i.e., a `node(`t_1`,`t_2`)` compound, where the two arguments – t_1 and t_2 – are both *itrees.*

We will refer to this data structure simply as a *tree* or as a *binary tree.*

1. Calculating tree depth

   The depth of a tree is the maximal number of `node(...)` structures nested into each other.

```
% tree_depth(+Tree, ?D): Tree is a binary tree of depth D.

| ?- tree_depth(leaf(3), D).
D = 0 ? ; no
| ?- tree_depth(leaf(5), 1).
no
| ?- tree_depth(node(node(leaf(1),leaf(4)),node(leaf(2),leaf(3))), D).
D = 2 ? ; no
| ?- tree_depth(node(leaf(1),node(leaf(2),node(leaf(4),leaf(3)))), D).
D = 3 ? ; no
```

   Hint: you can use the function `max` in arithmetic expressions, e.g.

```
| ?- X is max(2,3)+1. ---> X = 4 ? ; no
```

2. Checking the depth of the leaves of a tree
   The depth of a leaf is the number of nodes from the root to the leaf. In this problem we call a tree *depth tree*, if for all leaves, the value contained in the leaf is the depth of the leaf. Write a predicate to check if a thee is a depth tree.

```
% depth_tree(+Tree): Tree is a binary depth tree.

| ?- depth_tree(leaf(0)).
yes
| ?- depth_tree(node(node(leaf(2),node(leaf(3),leaf(3))),leaf(1))).
yes
| ?- depth_tree(node(node(leaf(2),node(leaf(3),leaf(4))),leaf(1))).
no
```

3. Simplifying linear expressions
   In this problem an *expression* is either a number, or the atom `x`, or a structure built from these using the binary operators `+`, `-`, and `*`. A *simple linear expression* is an expression in which at least one of the operands of each `*` operator can be simplified to a number using the usual algebraic transformations. For example, `(x-(x+1)*2-(1-x))*x` is a simple linear expression, but `(x+1)*(x+1)-x*x` is not.
   A simple linear expression can be transformed to the form a`*x+`b, where a and b are numbers (possibly zeros). This is called the simplified form of the simple linear expression.

```
% simplify(+U, ?SU): SU is the simplified form of the simple linear expression U.

| ?- simplify(((x+1)*3)+x+2*(x+x+3), S).
S = 8*x+9 ? ; no
| ?- simplify(2*3+x, S).
S = 1*x+6 ? ; no
| ?- simplify(((x+2)*3-2*x-(x+4))*(x+2*x+7)-9, S).
S = 6*x+5 ? ; no
| ?- simplify((x+1)*(x+1)-x*x, S).
no
```