Implement the predicate list_of(Elt, List) such that every element of
List is (equal to) Elt. What modes make sense for this predicate?
What modes does it actually work in?

Hint: the structure of the code is very similar to that of
proper_list/1 from the lecture notes.

QUESTION 2

Implement the predicate all_same(List) such that every element of
List is identical. This should hold for empty and single element
lists, as well.

QUESTION 3

Implement the predicate adjacent(E1, E2, List) such that E1 appears
immediately before E2 in List. Implement it by a single call to
append/3. What modes should and does this work in?

QUESTION 4

Reimplement the adjacent(E1, E2, List) predicate as a recursive
predicate that calls no other predicate but itself.

Hint: the structure of the code is very similar to that of member/2
from the lecture notes.

QUESTION 5

Implement the predicate before(E1, E2, List) such that E1 and E2 are
both elements of List, where E2 occurrs after E1 on List.

QUESTION 6

Suppose we wish to represent a set of integers as a binary tree. We
can use the atom empty to represent an empty tree or node, and
tree(L,N,R) to represent a node with label N (an integer), and left
and right subtrees L and R. Naturally, we want N to be strictly
larger than any label in L and strictly smaller than any in R. The
tree need not be balanced. For example,

        tree(tree(tree(empty, 1, empty),
                  2,
                  tree(empty, 3, tree(empty, 4, empty))),
             5,
             tree(tree(tree(empty,6,empty),
                       7,
                       tree(empty,8,empty)),
                  9,
                  tree(empty, 10, empty)))

is one possible representation of the set of numbers from 1 to 10. It
might be visualized as

            5
      /--------/ \--------\
      2   9
     / \   / \
    1   3   7   10
         \   / \
          4   6   8
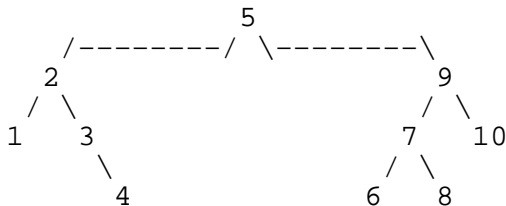
Hint: Prolog's arithmetic comparison operators are <, >, =< (not <=),
and >=. You can also use = and \= for equality and disequality.

Write a predicate intset_member(N, Set) such that N is a member of
integer set Set. Do not search in parts of the tree where the sought
element cannot be. This only needs to work when N is bound to an

other modes.

Hint: write one clause for the element being at the root of the
tree, one for it being in the left subtree, and one for the right subtree.

Write a predicate intset_insert(N, Set0, Set) such that Set is the
same as Set0, except that Set has N as a member. It doesn't matter
whether Set0 already has N in it, but Set must not have multiple
occurrences of N.