USING AND WRITING THE MAP FUNCTION (30/30 points)

The idea of this exercise is to use the principle of the `map` function to implement algorithms that transform data structures using higher-order functions.

1. Using the function `map` from the module `List`, write a function `wrap : 'a list -> 'a list list` that transforms a `list` of elements `'a` into a list of singleton lists.
   For instance, `wrap [1;2;3]` is equal to `[[1];[2];[3]]`
2. Consider the definition of the type `tree` given in the prelude. It represents binary trees carrying data items, on its internal nodes, and on its leaves.
   Write a function `tree_map : ('a -> 'b) -> 'a tree -> 'b tree` such that `tree_map f t` yields a tree of the same structure as `t`, but with all its data values `x` replaced by `f x`
   For example, suppose a function `string_of_int : int -> string`, that takes an integer and generates the string that represent this integer. Applied to `tree_map` and a tree of integers (i.e. of type `int tree`), it would yield a tree of strings (i.e. of type `string tree`).

THE GIVEN PRELUDE

```
type 'a tree =
    Node of 'a tree * 'a * 'a tree
  | Leaf of 'a;;
```

YOUR OCAML ENVIRONMENT

```
let wrap l =
  List.map (function x -> [x]) l
;;


let rec tree_map f = function
  | Leaf a -> Leaf (f(a))
  | Node (l, a, r) -> Node (tree_map f l, (f(a)), tree_map f r)
;;

```

Evaluate >

Switch >>

Typechecl

Reset Templ

Full-screen |

Check & Sa

Exercise complete (click for details) 30 pts

v Exercise 1: wrap Completed, 10 pts

Found wrap with compatible type.

Computing wrap [-2; -5; -4; -2; 1; -5; -4; -5; -5]

Correct value [[-2]; [-5]; [-4]; [-2]; [1]; [-5]; [-4]; [-5]; [-5]] 1 pt

Computing wrap [-1; 1; -1; 0; 4; 2; 3; -1]

Correct value [[-1]; [1]; [-1]; [0]; [4]; [2]; [3]; [-1]] 1 pt

Computing wrap [-2; -1; -3; 3; -5; 3; 2; 1]

Correct value [[-2]; [-1]; [-3]; [3]; [-5]; [3]; [2]; [1]] 1 pt

Computing wrap [-4; -4]

Correct value [[-4]; [-4]] 1 pt

Computing wrap [-1; 4; 1; 0]

Correct value [[-1]; [4]; [1]; [0]] 1 pt

```
Computing wrap ["ba-#ba , be "; "# baOCamlOCaml"; "be-, OCaml "]
Correct value [["ba-#ba , be "]; ["# baOCamlOCaml"]; ["be-, OCaml "]]        1 pt
Computing
  wrap
    [""; "bebe"; "OCP////#baOCPOCaml"; "--4456"; ", 4456-4456//, ";
     "-OCP 4456OCPba"; ""; "  #// "]
Correct value                                                               1 pt
  [[""]; ["bebe"]; ["OCP////#baOCPOCaml"]; ["--4456"]; [", 4456-4456//, "];
   ["-OCP 4456OCPba"]; [""]; ["  #// "]]
Computing wrap ["//4456"; "4456OCamlOCP////"; "OCP#OCP, -OCP"; ", "]
Correct value [["//4456"]; ["4456OCamlOCP////"]; ["OCP#OCP, -OCP"]; [", "]]  1 pt
Computing wrap ["OCP"; "# OCaml"; "//babe4456"]
Correct value [["OCP"]; ["# OCaml"]; ["//babe4456"]]                        1 pt
v Exercise 2: tree_map                                          Completed, 20 pts
Found tree_map with compatible type.
Computing tree_map <fun> (Leaf 4)
Correct value (Leaf 46)                                                     1 pt
Computing
  tree_map
    <fun>
    (Node (Node (Node (Leaf (-3), 1, Leaf 1), -5, Leaf (-3)), -2,
      Node (Leaf (-3), 4, Leaf (-3))))
Correct value                                                               1 pt
  (Node (Node (Node (Leaf 6, 2, Leaf 2), 8, Leaf 6), 5,
    Node (Leaf 6, -1, Leaf 6)))
Computing
  tree_map
    <fun>
    (Node
      (Node
        (Node (Leaf (-4), -1,
          Node
           (Node (Leaf (-3), 2,
             Node (Node (Leaf 3, 1, Node (Leaf (-2), -2, Leaf (-5))), 4,
              Leaf (-1))),
           3, Node (Leaf 4, -4, Leaf 4))),
        -5, Leaf (-5)),
      -2, Leaf (-4)))
Correct value                                                               1 pt
  (Node
    (Node
      (Node (Leaf (-8), -2,
        Node
         (Node (Leaf (-6), 4,
           Node (Node (Leaf 6, 2, Node (Leaf (-4), -4, Leaf (-10))), 8,
            Leaf (-2))),
         6, Node (Leaf 8, -8, Leaf 8))),
      -10, Leaf (-10)),
    -4, Leaf (-8)))
Computing tree_map <fun> (Leaf (-4))
Correct value (Leaf 7)                                                      1 pt
Computing
  tree_map
    <fun>
    (Node (Leaf (-4), 3,
      Node (Node (Leaf 0, -4, Node (Node (Leaf 1, -5, Leaf (-5)), 0, Leaf 2)),
       -4, Leaf 2)))
Correct value                                                               1 pt
  (Node (Leaf 38, 45,
    Node (Node (Leaf 42, 38, Node (Node (Leaf 43, 37, Leaf 37), 42, Leaf 44)),
     38, Leaf 44)))
Computing
  tree_map
    <fun>
    (Node (Leaf 2, 0,
      Node (Leaf 4, -2, Node (Node (Leaf (-1), -1, Leaf (-2)), -1, Leaf 0))))
Correct value                                                               1 pt
  (Node (Leaf 4, 0,
    Node (Leaf 8, -4, Node (Node (Leaf (-2), -2, Leaf (-4)), -2, Leaf 0))))
Computing tree_map <fun> (Node (Leaf 1, 4, Node (Leaf 2, -2, Leaf (-5))))
Correct value (Node (Leaf 43, 46, Node (Leaf 44, 40, Leaf 37)))            1 pt
Computing tree_map <fun> (Leaf 4)
Correct value (Leaf (-1))                                                   1 pt
Computing tree_map <fun> (Leaf 3)
Correct value (Leaf 6)                                                      1 pt
Computing
```

```
Correct value (Node (Node (Node (Leaf [illegible]                            1 pt
Found tree_map with compatible type.
Computing tree_map <fun> (Leaf (-1))
Correct value (Leaf true)                                                    1 pt
Computing
  tree_map
    <fun>
    (Node (Node (Node (Leaf 2, 3, Leaf 1), -5, Leaf 0), 0, Leaf 1))
Correct value                                                                1 pt
  (Node (Node (Node (Leaf true, false, Leaf true), true, Leaf true), true,
    Leaf true))
Computing tree_map <fun> (Leaf 4)
Correct value (Leaf false)                                                   1 pt
Computing tree_map <fun> (Leaf (-2))
Correct value (Leaf true)                                                    1 pt
Computing tree_map <fun> (Leaf 2)
Correct value (Leaf true)                                                    1 pt
Computing
  tree_map
    <fun>
    (Node (Leaf 2, -5, Node (Node (Leaf (-4), 4, Leaf 1), 1, Leaf 4)))
Correct value                                                                1 pt
  (Node (Leaf false, false,
    Node (Node (Leaf false, false, Leaf false), false, Leaf false)))
Computing tree_map <fun> (Leaf (-1))
Correct value (Leaf false)                                                   1 pt
Computing tree_map <fun> (Node (Leaf (-1), 2, Leaf 3))
Correct value (Node (Leaf false, false, Leaf false))                         1 pt
Computing tree_map <fun> (Leaf 2)
Correct value (Leaf false)                                                   1 pt
Computing tree_map <fun> (Leaf (-5))
Correct value (Leaf true)                                                    1 pt
```