Homework 5

Submission is on this page (https://canvas.umn.edu/courses/391238/assignments/3485981) .

In this assignment, you will be implementing 5 functions, 4 of which are in a module interfacing with the following binary tree type with integer values:

```
type tree =
  | Leaf
  | Node of int * tree * tree
```

Submission: hw5.ml

- **hw5.ml**: Should contain definitions of `with_default`, and `Tree_ops` module functions `sum`, `tmax`, `flatten`, and `is_tree_sorted`. **Template can be found here (https://canvas.umn.edu/courses/391238/files/38469775?wrap=1)** ↓ **(https://canvas.umn.edu/courses/391238/files/38469775/download?download_frd=1)** .

P1) `with_default : int -> int option -> int`

This function strips an `int option` type of the `Some` constructor. If the input `int option` is `None`, return the `int` input as the default. Examples:

```
with_default (-1) None = -1
```

```
with_default (-1) (Some 7) = 7
```

P2) `sum : tree -> int`

This function is defined within the `Tree_ops` module. The input is a `tree` type, and the output is the sum of all nodes in the tree. Examples:

```
Tree_ops.sum Leaf = 0
```

```
Tree_ops.sum (Node (7, Node (8, Leaf, Leaf), Node (-9, Leaf, Leaf))) = 6
```

P3) `tmax : tree -> int option`

This function returns the maximum value in the tree in the form of an `int option` type. If the tree is a `Leaf`, then return `None`. *Hint: function* `with_default` *can be useful here.*

```
Tree_ops.tmax Leaf = None
```

```
Tree_ops.tmax (Node (7, Node (8, Leaf, Leaf), Node (-9, Leaf, Leaf))) = Some 8
```

P4) `flatten : tree -> int list`

This function returns the flattened tree as an `int list`. For `Node (x, tree_l, tree_r)`, values in `tree_l` will always be left of `x` in the list, and values in `tree_r` will always be right of `x` in the list. Examples:

```
Tree_ops.flatten Leaf = []
```

```
Tree_ops.flatten (Node (7, Node (8, Leaf, Leaf), Node (-9, Leaf, Leaf))) = [8; 7; -9]
```

```
Tree_ops.flatten (Node (1, Node (21, Node (31, Leaf, Leaf), Node (32, Leaf, Leaf)), Node (22, Node (33, Leaf, Leaf), Node (34, Leaf, Leaf))) = [31; 21; 32; 1; 33; 22; 34]
```

P5) `is_tree_sorted : tree -> bool`

Let `tmax tree`, `tmin tree` be the maximum and minimum values of the tree, if any. A tree is sorted if for all subtrees `Node (x, tree_l, tree_r)` in the tree, `tmax tree_l` $\leq$ `x` $\leq$ `tmin tree_r`. Hint: the `flatten` function may be helpful. Examples:

```
Tree_ops.is_tree_sorted Leaf = true
```

```
Tree_ops.is_tree_sorted (Node (7, Node (8, Leaf, Leaf), Node (-9, Leaf, Leaf))) = false
```

```
Tree_ops.is_tree_sorted (Node (2, Node (2, Leaf, Leaf), Node (2, Leaf, Leaf))) = true
```

```
Tree_ops.is_tree_sorted (Node (35, Node (12, Node (7, Leaf, Leaf), Node (35, Leaf, Leaf)), Node (48, Leaf, Leaf))) = true
```