Assignment 12.4 (H) Len or nlen? [5 Points]

The following functions are defined:

```
let rec nlen n l = match l with [] -> 0
  | h::t -> n + nlen n t

let rec fold_left f a l = match l with [] -> a
  | h::t -> fold_left f (f a h) t

let rec map f l = match l with [] -> []
  | h::t -> f h :: map f t

let (+) a b = a + b
```

Show that the statement

```
nlen n l = fold_left (+) 0 (map (fun _ -> n) l)
```

holds for arbitrary `l` and `n`. Assume that all expressions do terminate.

Suggested Solution 12.4

We have to prove the more general statement:

```
acc + nlen n l = fold_left (+) acc (map (fun _ -> n) l)
```

We do so by induction on the length k of list `l`.

- Base case: $k = 0$, so `l = []`

```
    acc + nlen n []
=nlen acc + match [] with [] -> 0 | h::t -> n + nlen n t
=match acc + 0
=arith acc
=match match [] with [] -> acc | h::t -> fold_left (+) ((+) acc h) t
=f_l fold_left (+) acc []
=match fold_left (+) acc (match [] with [] -> []
        | h::t -> (fun _ -> n) h :: map (fun _ -> n) t)
=map fold_left (+) acc (map (fun _ -> n) [])
```

- Inductive step: We assume the statement holds for a list `l = xs` of length $k \geq 0$.

Now, we prove it for `l = x::xs`:

```
    acc + nlen n (x::xs)
nlen
=   acc + match x::xs with [] -> 0 | h::t -> n + nlen n t
match
=   acc + n + nlen n xs
I.H.
=   fold_left (+) (acc + n) (map (fun _ -> n) xs)
(+)
=   fold_left (+) ((+) acc n) (map (fun _ -> n) xs)
match
=   match n::map (fun _ -> n) xs with [] -> acc
        | h::t -> fold_left (+) ((+) acc h) t
f_l
=   fold_left (+) acc (n :: map (fun _ -> n) xs)
fun
=   fold_left (+) acc ((fun _ -> n) x :: map (fun _ -> n) xs)
match
=   fold_left (+) acc (match x::xs with [] -> []
        | h::t -> (fun _ -> n) h :: map (fun _ -> n) t)
map
=   fold_left (+) acc (map (fun _ -> n) (x::xs))
```

□