Ihr Repetitoriums-Tutor Kevin ~~Kappelmann~~ OCamlmann hatte Ihnen einst von der Theorie "Everything is a fold" erzählt. Wir wollen dieser Theorie nun auf den Grund gehen und uns davon überzeugen.

Gegeben sind folgende OCaml-Definitionen:

let rec fold_left f a = function [] → a | x::xs → fold_left f (f a x) xs
let rec map f = function [] → [] | x::xs → f x::map f xs
let rec append l1 l2 = match l1 with [] → l2 | x::xs → x::append xs l2
let (@) = append
let rec reverse = function [] → [] | x::xs → reverse xs @ [x]
let rec sum i = function [] → i | x::xs → sum (x+i) xs

Ihr Tutor gibt Ihnen des Weiteren folgende zwei Lemmata, die er einst auf magische Art und Weise bewiesen hat:

Lemma1: append l1 [] = l1
Lemma2: append (append l1 l2) l3 = append l1 (append l2 l3)

1) Zeigen Sie für passende Funktionen f und alle Listen l:
   reverse (fold_left (fun a x → f x::a) [a] l) = reverse [a] @ (map f l)
   (8 Punkte)

2) Zeigen Sie nun für passende Funktionen f und alle Listen l:
   reverse (fold_left (fun a x → f x::a) [] l) = map f l
   (2 Punkte)

3) Mit voller Stolz erkennen Sie nun, dass sich jeder map Aufruf als ein fold darstellen lässt, doch damit nicht genug! Beweisen Sie für alle int-Listen l und Ganzzahlen i:
   fold_left (fun a x → x+a) i l = sum i l

   Worin läge der Unterschied zum Beweis von
   fold_left (fun a x → x+a) 0 l = sum 0 l
   Was für Probleme gäbe es im Beweis?
   (4 Punkte)

Sie können im folgenden davon ausgehen, dass alle Aufrufe terminieren.
Ergebnisse aus den vorherigen Teilaufgaben dürfen als gegeben betrachtet werden.
Geben Sie für jeden Schritt die verwendete Umformungsregel an (Def. <name>, Lemma 1, I.H., Aufgabe 1,….).