## 12.4

```
z.Z.:

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

## IB

```
z.Z:

nlen n []= fold_left (+) 0 (map (fun _ -> n) [])

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

## IA

```
m+nlen n xs=fold_left (+) m (map (fun _ -> n) xs)
```

## IS

```
z.Z: m+nlen n x::xs=fold_left (+) m (map (fun _ -> n) x::xs)

m+nlen n x::xs
(nlen)=m+match x::xs with [] -> 0 | h::t -> n+nlen n t
(match)=m+n+nlen n xs
(arith)=n+m+nlen n xs
(ia)=n+fold_left (+) m (map (fun _ -> n) xs)
(arith)=(+) n (fold_left (+) m (map (fun _ -> n) xs))
(fold_left)=(+) n (match (map (fun _ -> n) xs) with [] -> m
         | h::t -> fold_left (+) ((+) m h) t)
...
=fold_left (+) ((+) m n) (map (fun _ -> n) xs)
(map)=match n::(map (fun _ -> n) xs) with [] -> m
         | h::t fold_left (+) ((+) m h) t
(fold_left)=fold_left (+) m ((fun _ -> n) x :: map (fun _ -> n) xs)
(match)=fold_left (+) m (match x::xs with [] -> []
          | h::t -> (fun _ -> n) h :: map (fun _ -> n) t))
(map)=fold_left (+) m (map fun _ -> n) x::xs)
```

### TODO

```
n+fold_left (+) m (map (fun _ -> n) xs)=
fold_left (+) (m+n) (map (fun _ -> n) xs)
```

## 12.5

### IB

```
z.Z: fl (+) 0 (rev_map (fun x -> x*2) l []) = fr (fun x a -> a+2*x) l 0

fl (+) 0 (rev_map (fun x -> 2*x) [] [])
=fl (+) 0 (match [] with [] -> []
          | x::xs -> rev_map (fun x -> 2*x) xs ((fun x -> 2*x ) x::[]))
=fl (+) 0 []
=match [] with [] -> 0
=0
=match [] with [] -> 0
          | x::xs (fun x a -> a+2*x) x (fr (fun x a -> a+2*x) xs 0)
=fr (fun x a -> a+2*x) [] 0
```

### IA

```
fl (+) 0 (rev_map (fun x -> x*2) xs []) = fr (fun x a -> a+2*x) xs 0
```

### IS

```
z.Z: fl (+) 0 (rev_map (fun x -> x*2) h::t []) = fr (fun x a -> a+2*x) h::t 0

fl (+) 0 (rev_map (fun x -> x*2) h::t [])
=match
          (rev_map (fun x -> x*2) h::t [])
          with [] -> 0 | x::xs -> fl (fun x -> x*2) ((fun x -> x*2) 0 x) xs
=match
          (match h::t with [] -> []
                    | x::xs -> rev_map (fun x -> x*2) xs ((fun x -> x*2) x::[]))
          with [] -> 0 | x::xs -> fl (fun x -> x*2) ((fun x -> x*2) 0 x) xs
=match
          (rev_map (fun x -> x*2) t ((fun x -> x*2) h::[]))
```

```
        with [] -> 0 | x::xs -> fl (fun x -> x*2) ((fun x -> x*2) 0 x) xs
=match
        (rev_map (fun x -> x*2) t (2*h::[])
        with [] -> 0 | x::xs -> fl (fun x -> x*2) ((fun x -> x*2) 0 x) xs
...
=fr (fun x a -> a+2*x) h::t 0
```

## 12.6

```
z.Z: fl (+) 0 (to_list t)=tf add3 0 t
```

### IB

```
z.Z: fl (+) 0 (to_list Empty)=tf add3 0 Empty

fl (+) 0 (to_list Empty)
(to_list)=fl (+) 0 (match Empty with Empty -> []
        | Node (x,l,r) -> app (to_list l) (x::to_list r)
(match)=fl (+) 0 []
(fl)=match l with [] -> 0
        | x::xs -> fl (+) ((+) 0 x) xs
(match)=0
(match)=match Empty with Empty -> 0
        | Node (x,l,r) -> add3 (tf add3 0 l) (x::to_list r)
(tf)=tf add3 0 Empty
```

### IA

```
fl (+) m (to_list t)=tf add3 m t
```

### IS

```
z.Z: fl (+) m (to_list Node(v,t1,t2))=tf add3 m Node(v,t1,t2)

fl (+) m (to_list Node(v,t1,t2))
(to_list)=fl (+) m (match (Node(v,t1,t2) with Empty -> []
        | Node (x,l,r) -> app (to_list l) (x::to_list r)
(app)=fl (+) m (match t with Empty -> []
        | Node (x,l,r) ->

=(fl (+) m (to_list t1)+v+(match (to_list t2) with [] -> m
        | x::xs -> fl x::
```

```
=fl (+) m (to_list t1)+v+fl (+) m (to_list t2)
(ia)=(tf add3 m t1)+v+(tf add3 m t2)
(add3)=add3 (tf add3 m t1) v (tf add3 m t2)
(match)=match Node(v,t1,t2) with Empty -> m
         | Node(x,l,r) -> add3 (tf add3 m l) x (tf add3 m r)
(tf)=tf add3 m Node(v,t1,t2)
```

## TODO

```
n+fl (+) m l=fl (+) (m+n) l
```