2.b)
=> (alternating-sum (list 1 2 3 4 5))
=> (helper (list 1 2 3 4 5) 0 0)
=> (helper (list 2 3 4 5) 1 1)
=> (helper (list 3 4 5) 0 -1)
=> (helper (list 4 5) 1 2)
=> (helper (list 5) 0 -2)
=> (helper (empty) 1 3)
=> 3

2.c)
Using tail recursion here is a lot more efficient than using plain recursion, as with plain recursion every function call is accompanied by carry-on from the previous call causing a large expression to build up as we go deeper in the evaluation tree, all that build-up will have to be stored in memory until we reach the end of the tree where it gets evaluated, for a larger tree this can be problematic and possibly cause an overflow.

3)
(f, 3)
(* (f (dec (dec 3))) (f (dec 3)))
(* (f (dec (- 3 1))) (f (- 3 1)))
(* (f (dec 2)) (f 2))
(* (f (- 2 1)) (f 2))
(* (f 1) (f 2))
(* (- 10 1) (- 10 2))
(* 9 8)
72