```
type nat = Zero | S of nat
module Nat_ops =

struct
  let rec nat_of_int i =
    if i <= 0 then
        Zero
    else
        S (nat_of_int (i-1))
  let rec int_of_nat n =
    match n with
    | Zero -> 0
    | S m -> 1 + (int_of_nat m)
  let rec plus a b =
    match b with
    | Zero -> a
    | S c -> plus (S a) c
end
----------------------------------------------------------------------------
lemma:
prove: plus (S a) b = S(plus a b)

base case: given that b == Zero; plus (S a) b = S(plus a b)

    plus (S a) b
= { case }
    plus (S a) Zero
= { plus def }
    match Zero with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    S(a)
= { reverse match }
    S(  match Zero with
    | Zero -> a
    | S c -> plus (S a) c)
= { apply plus def }
    S(plus a Zero)
= { case }
    S(plus a b)

Induction step: plus (S a) (S b)
Induction hypothesis (IH): given that b == (S b); plus (S a) b = S(plus a b)

    plus (S a) b
= { case }
    plus (S a) (S b)
= { plus def }
    match (S b) with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    plus (S (S a)) b
= { IH }
    S(plus a b)

P1) Prove plus Zero b = b

Base case: given b = Zero, proof plus Zero b = b

    plus Zero b
= { case }
    plus Zero Zero
= { plus def }
    match Zero with
    | Zero -> a
```

```
    | S c -> plus (S a) c
= { apply match }
    Zero
= { case }
    b

Induction step: plus Zero (S b)
Induction hypothesis (IH): plus Zero b = b

    plus Zero b
= { case }
    plus Zero (S b)
= { plus def }
    match (S b) with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    plus (S Zero) b
= { apply lemma }
    S(plus Zero b)
= { S( apply (IH) )}
    S(b)
= { case }
    b

P2) Prove plus a b = plus b a
Base case: Given any a, prove that for b = Zero statement holds true

    plus a b
= { case }
    plus a Zero
= { plus def }
    match Zero with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    a
= { apply reverse lemma P1:  plus Zero b = b}
    plus (S Zero) a
= { reverse match }
    match (S a) with
    | Zero -> a
    | S c -> plus (S a) c
= { plus def }
    plus Zero (S a)
= { case }
    plus b a

Induction: Given any a, assume statement holds for b. Show for S b
IH: for all a, plus a b = plus b a

= ...
    plus a b
= { case }
    plus a (S b)
= { plus def }
    match (S b) with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    plus (S a) b
= {IH}
    plus b (S a)
= { apply plus }
    match (S a) with
    | Zero -> a
    | S c -> plus (S a) c
```

```
= { plus def }
    plus (S b) a
= { case }
    plus b a


P3) Prove plus a (plus b c) = plus (plus a b) c

base case: c = Zero
    plus a (plus b c)
= { case }
    plus a (plus b Zero)
= { plus def }
    plus a ( match Zero with
            | Zero -> a
            | S c -> plus (S a) c
    )
= { apply match }
    plus a b
= { reverse match }
    match Zero with
    | Zero -> a
    | S c -> plus (S a) c
= { plus def }
    plus (plus a b) Zero
= { case }
    plus (plus a b) c

Induction : given any a and b, let c = (S c)
{IH} : plus a (plus b c) = plus (plus a b) c

    plus a (plus b c)
= { case }
    plus a (plus b (S c))
= { plus def }
    plus a (    match (S c) with
            | Zero -> a
            | S c -> plus (S a) c
    )
= { apply match }
    plus a (plus (S b) c)
= { apply lemma }
    plus a S(plus b c)
= { plus def }
    match S(plus b c) with
    | Zero -> a
    | S c -> plus (S a) c
= { apply match }
    plus (S a) (plus b c)
= { apply lemma }
    S(plus a (plus b c))
= { IH }
    S(plus (plus a b) c)
= { apply lemma }
    plus S(plus a b) c
= { reverse match }
    match (S c) with
    | Zero -> a
    | S c -> plus (S a) c
= { plus def }
    plus (plus a b) (S c)
= { case }
    plus (plus a b) c
```