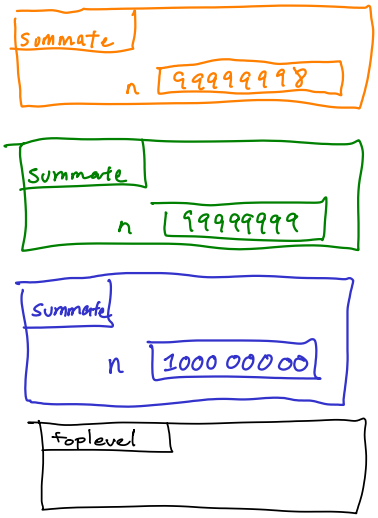
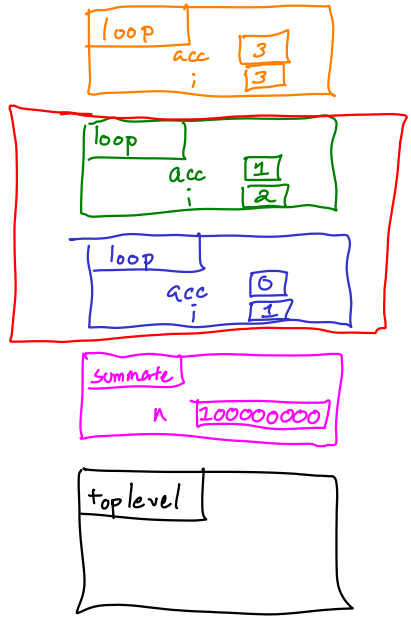
```
let rec summate n =
    if n = 1 then 1 else
     n + summate (n-1)
;;

summate 100000000
```




```
let rec summate n =
    if n = 1 then 1 else
     n + summate (n-1)
;;
```

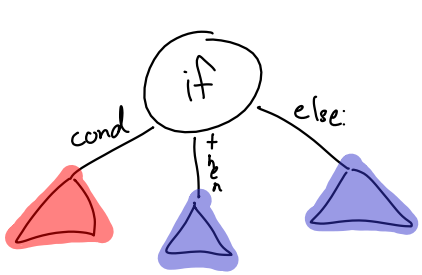
```
(100000000+ (99999999+ (if 99999998 = 1 then 1 else
 99999998 + summate (99999998 -1))))
```

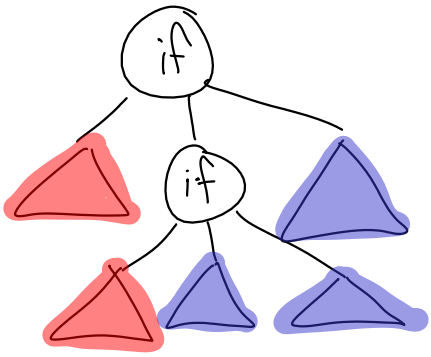
```
let summate n =
   let rec loop acc i =
     if i = n+1 then
       acc
     else
       loop (acc+i) (i+1)
   in
   loop 0 1
;;
Summate 100000000
```



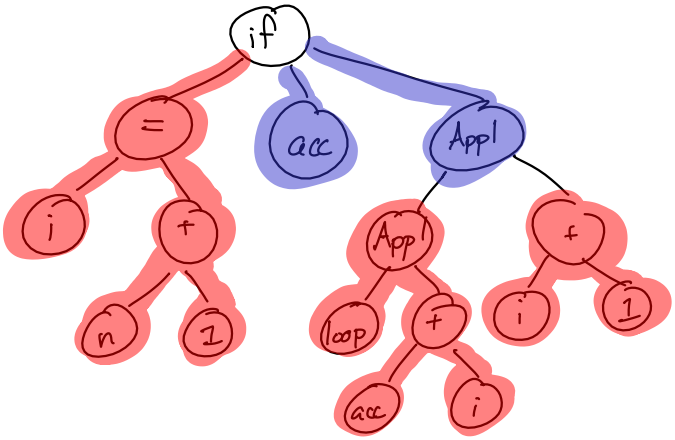

```
let rec loop acc i =
  if i = 100000000 +1 then
    acc
  else
    loop (acc+i) (i+1)
in
if 3 = 100000000 +1 then
   3
else
   loop (3+3) (3+1)
```

A subexpression is a "tail expression" of a larger expression if the tail expression is the last thing the larger expression computes and the tail expression produces the result of the larger expression.






```
type expr =
  | EInt of int
  | EBool of bool
  | EUnaryOp of unary_operator * expr
  | EBinaryOp of binary_operator * expr * expr
  | ETuple of expr list
  | ELet of string * expr * expr
  | EVar of string
  | EIf of expr * expr * expr
  | EAppl of expr * expr
  | ESet of expr * expr * expr
```




```
let rec loop acc i =
  if i = n+1 then
    acc
  else
    loop (acc+i) (i+1)
in
```

"Tail call" is a call which is a tail expression