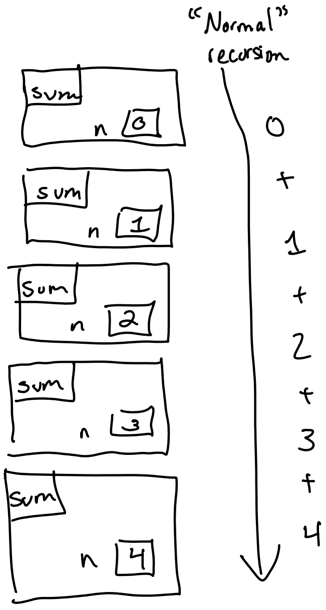
Tail Call Optimization

```
def loop n =
   if  n > 0  then
        print(loop(n-1))
   else
        0
end
loop 100
```

```
let rec sum n =
   if n>0 then
      n + (sum (n-1))
   else
      0
;;
sum 4
```

```
if 4>0 then
   4 + (sum (4-1)) => 4 + (sum (4-1)) => 4 +  if 3>0 then
                                                 3 + (sum (3-1)) =>
else                                           else
   0                                              0
```

```
4 + 3 + (sum (3-1)) => ... => 4 + 3 + 2 + 1 + 0
```



```
let sum n =
  let rec loop i acc =
    if i>n then
      acc
    else
      loop (i+1) (acc+i)
  in
  loop 0 0
;;
```

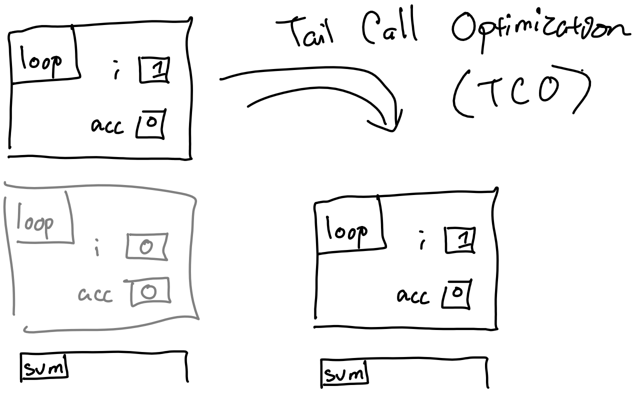
```
sum 4  ⟹  let rec loop i acc =
             if i>4 then
               acc
             else
               loop (i+1) (acc+i)
           in
           loop 0 0

⟹  if 0>4 then
      0
    else
      loop (0+1) (0+0)

⟹  loop (0+1) (0+0)

⟹  if 1>4 then
      0
    else
      loop (1+1) (0+1)

⟹  loop (1+1) (0+1)  ⟹  if 2>4 then
                            1
                          else
                            loop (2+1) (1+2)
```


Tail Call Optimization (TCO)

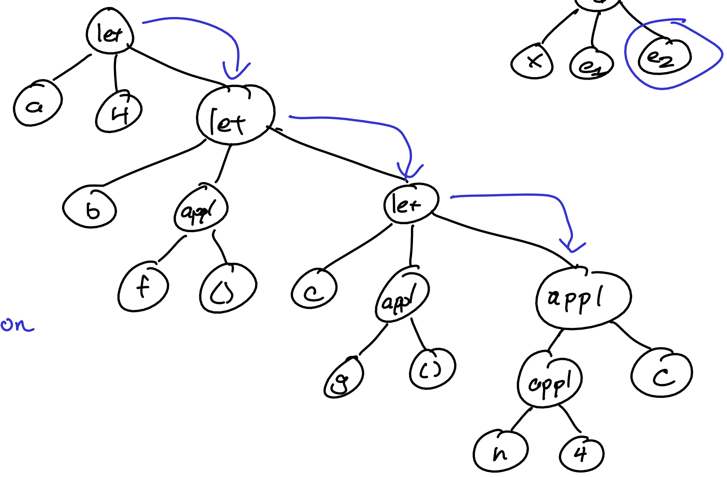
```
type c_expr =
    CIExpr of i_expr
  | CIf of i_expr * a_expr * a_expr
  | CTuple of i_expr list
  | CUnaryOp of unary_op * i_expr
  | CBinaryOp of binary_op * i_expr * i_expr
  | CAppl i_expr * i_expr
```

```
type a_expr =
  | ALet of string * a_expr * a_expr
  | ACExpr of c_expr
```

```
let x = f 0 in
g x
```

```
let a = 4 in
let b = f () in
let c = g () in
h 4 c
```

tail position



TCO only works if the last work to do is a function call

```
let rec fib n =
    if n=0 then 0
    else if n=1 then 1
    else (fib (n-1)) + (fib (n-2))
```

Given $\underline{e}$, can we TCO?

```
def f ....... =
    1+ (g 0)
end
```

```
(f (), g ())
```

```
if f 0 then
   g 3
else
   g 4
```