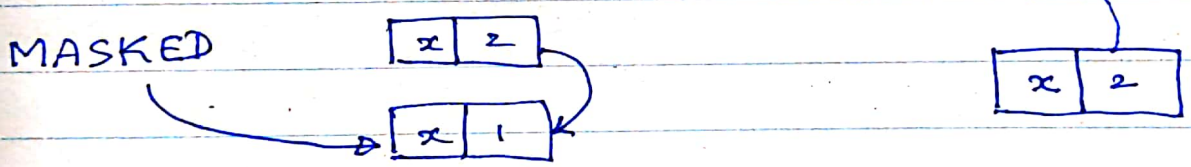
Names Values Expressions

Binding (Name, Value)

The meaning of a construct should not change - static binding
lexical scoping

An environment is a "structure" containing bindings

```
let x = 1;;
let x = 2;;
```

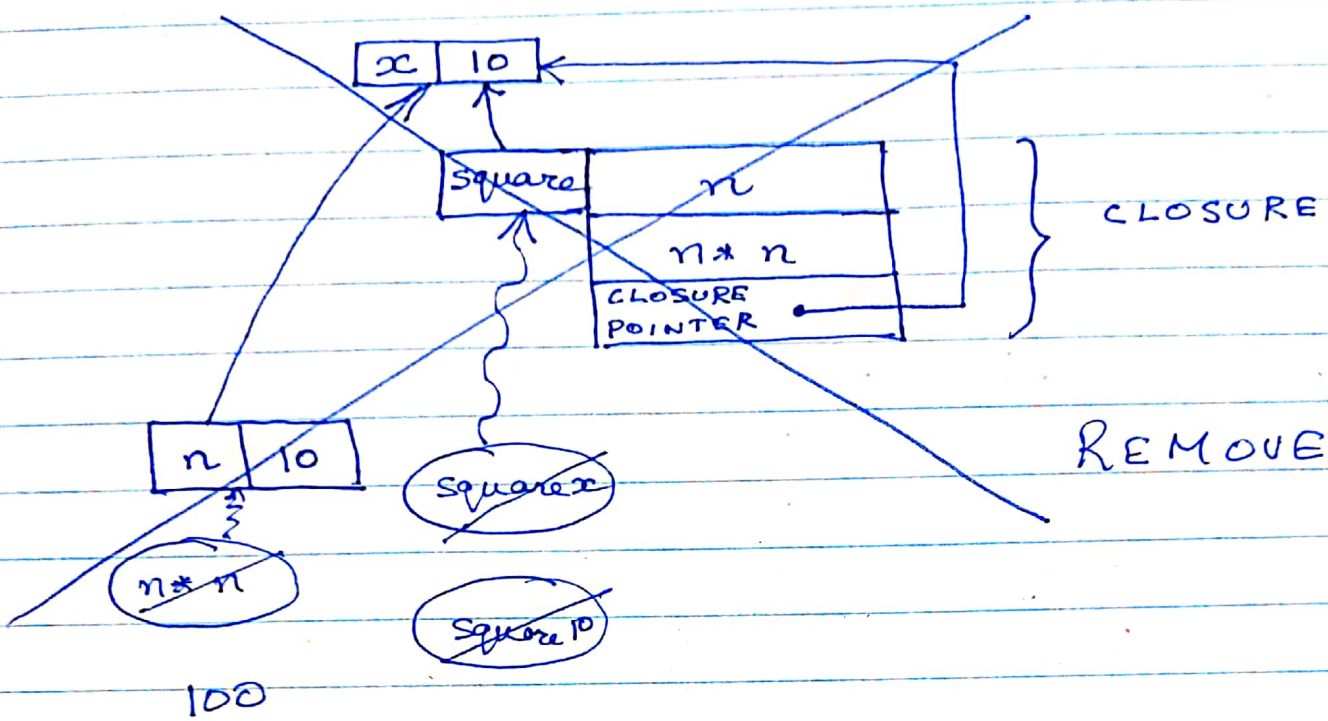


LOCAL BINDINGS

let name = exp_1 in exp_2

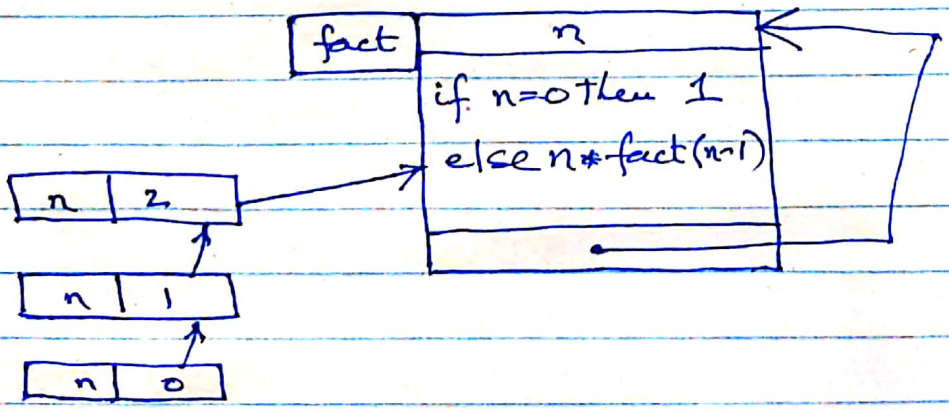
① Evaluate exp_1 to get v_1
② Bind name to v_1 & put it on top of the env.
③ Use this binding to evaluate exp_2
④ When this evaluation is complete remove the binding from ②

```
let x = 10 in
let square n = n * n in
square x;;
```

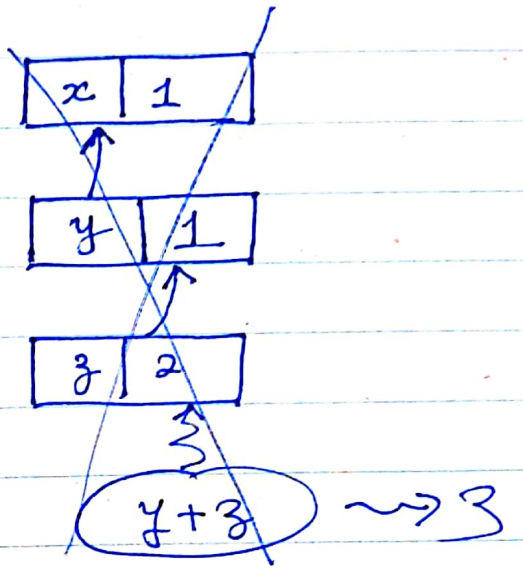


Recursive functions are special:
if you use let rec the closure pointer will include the frame that is being set up.

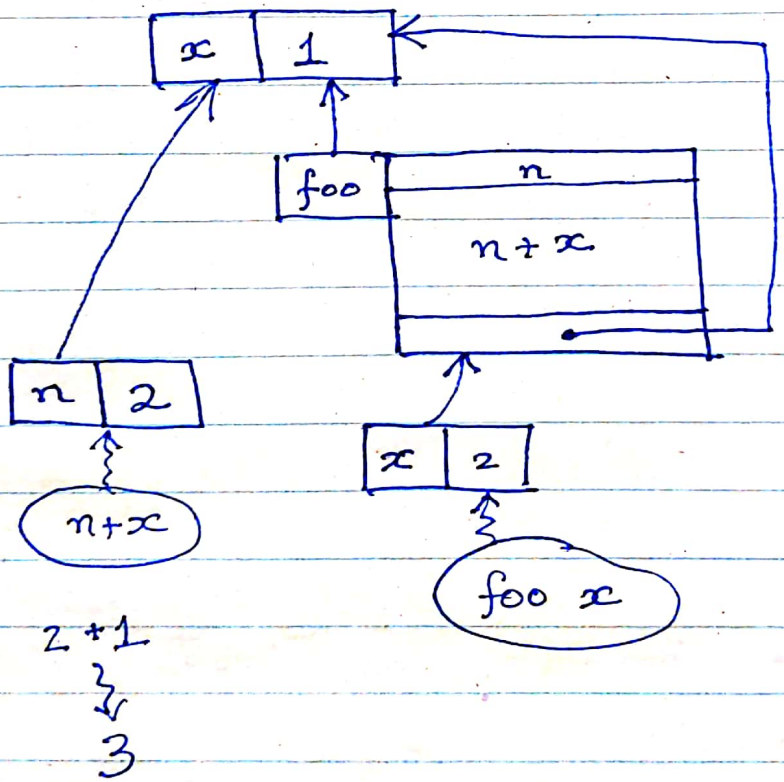
```
let rec fact n = if n=0 then 1 else n * fact (n-1)
```



```
let x = 1 in
  let y = x in
    let z = 2 in
      y+z;;
```

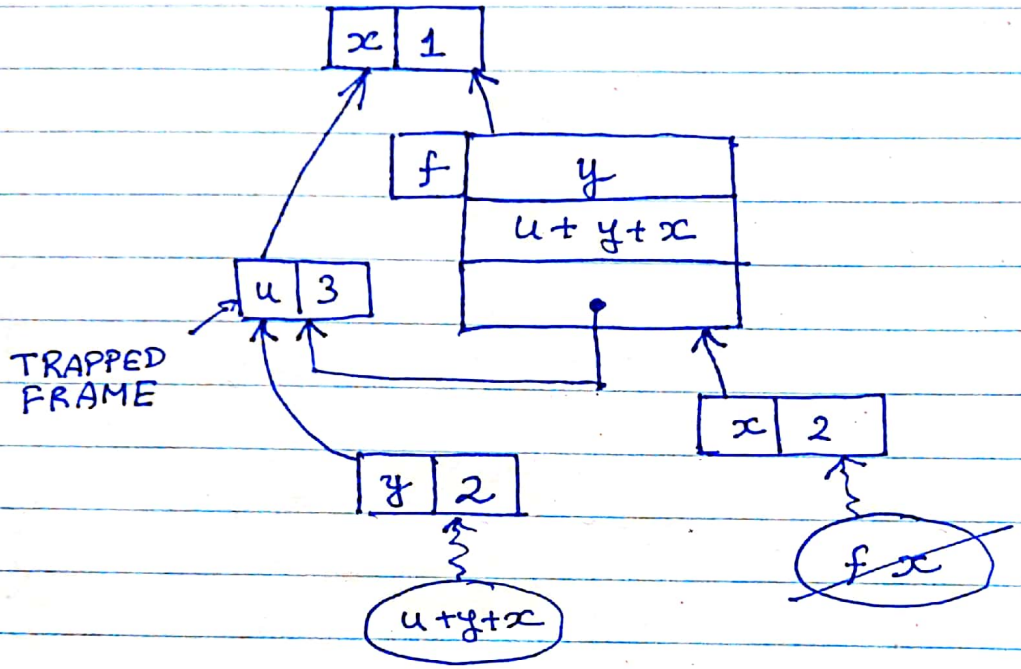



```
let x = 1 in
  let foo n = n+x in
    let x = 2 in
      foo x;;
```



2+1
3

```
let x = 1 in
  let f =
    ( let u = 3 in (fun y → u+y+x)) in
  let x = 2 in
    f x
```



3 + 2 + 1 ⇝ 6

```
let foo = - - - -
let bar = - - - foo - - -
```

foo has an error, it needs to be corrected.

```
let foo = - - - <CORRECTION> - - -
```

bar will remember the old definition.
You need to re-evaluate bar as well.

```
let myadd a b = a + b
≡ let myadd = fun a → (fun b → a + b)
```

```
let rec insert (n, l) =
    match l with
    | [] ⟶ [n]
    | x :: xs ⟶ if n ≤ x then n :: l
                  else x :: (insert n xs)
```

ASSUMPTION ; l is already sorted

WHAT WE WANT : output is also sorted.

Proof by induction on the length of l

Base case $l = [\,]$

output = $[n]$, is it sorted? Yes!

Inductive case Assume insert works correctly when $|l| \leq k$ for some $k \geq 0$.

Now we consider an l of length $k+1$

x is a number, $|xs| = k$.

output :

case (a) $n \leq x$, $n :: (x :: xs)$ is clearly sorted

case (b) $n > x$ $x :: (\text{insert } n \; xs)$

$x \leq$ anything in xs

$x \leq n$

(insert n xs) is sorted by inductive assumptio