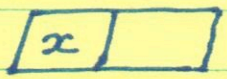
Rules for drawing environment diagrams:

1. Every "let" opens a new scope for bindings: so when you see

   let x = ...

   you should immediately draw

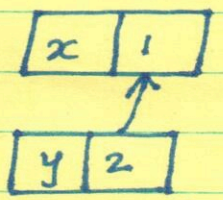


   with the box on the right empty at first. We will fill it in presently.

2. There should be an arrow from the new box to the NEAREST ENCLOSING box that is <u>still open</u>. So, if we have

   let x = 1 in
   let y = 2 in

   We draw



   The arrow goes from one <u>whole box</u> to another; in this picture the arrow is pointing at [x|1], <u>NOT</u> at [1]!!

3. To fill the value in the right-hand-side of a box you must produce a value: (a) an integer, (b) a boolean (c) a float (d) a string (e) a data structure OR (f) a function (together with some additional data). (g) a memory address [LATER]

(4) We need to evaluate the expression exp in

let x = exp

in order to get a value. This expression is to be evaluated in the environment that exists WHEN you enter the let. It cannot include the value you are defining. Thus you cannot implement recursion with a let.

(5) To implement a recursive definition you have to use the special keyword rec

let rec fact n = ...

This causes the new environment pointer to point to the frame being created.

(6) The evaluation of a function definition produces a CLOSURE. Thus

let foo n = n+1729

produces a closure, as does

let foo = fun n → n+1729

(7) What is a closure? : A closure has 3 pieces:

(a) a (list of) parameter(s)

(b) a body, i.e. code which may mention names

(c) a pointer to the environment that exists when the function is defined. You never follow this pointer when searching for bindings.

How a function is called:

(i) First, evaluate the argument(s) in the current environment.

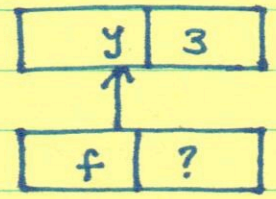
(ii) Create a new frame (binding) matching the parameter with the value produced by evaluating the argument.

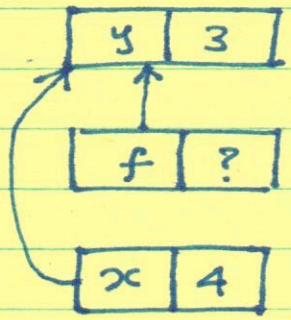
(iii) The pointer from this frame goes to the same place as the pointer in the closure.

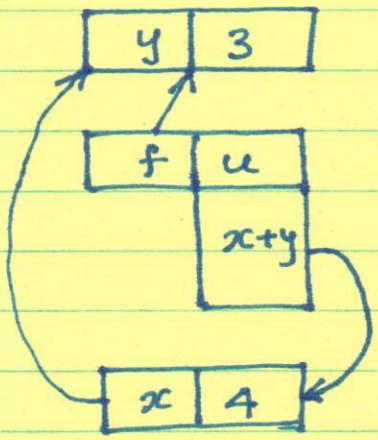
(iv) Now evaluate the body. In doing so you follow pointers until you find the name you need. You NEVER go inside closures and follow pointers there. The closure is used to set up the environment for evaluating the function body.

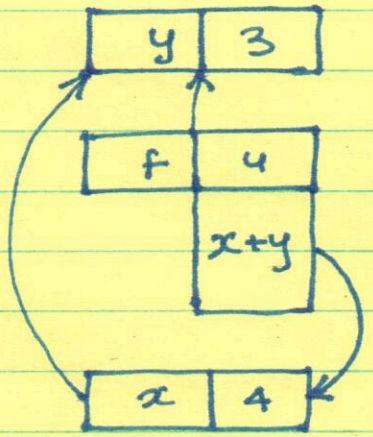
———— x ————

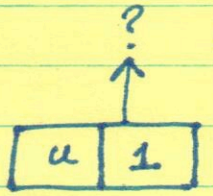
let y = 3 in
let f = let x = 4 in fun u → x+y in
f(1)





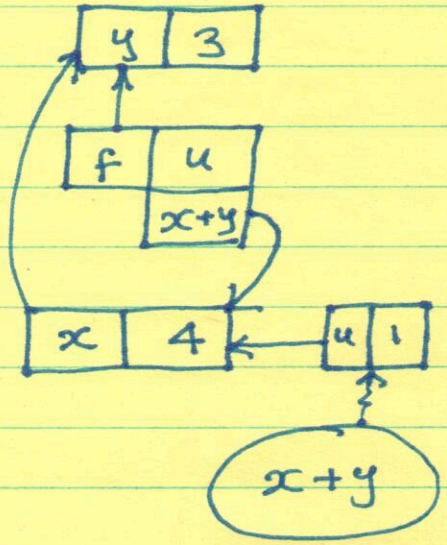






WHERE DOES THIS GO?

Ans: To the same place as the arrows in the closures



⇝ 7