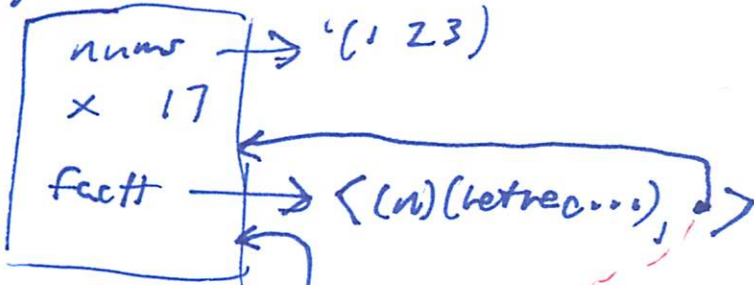
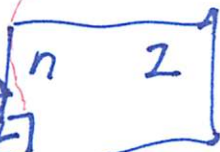


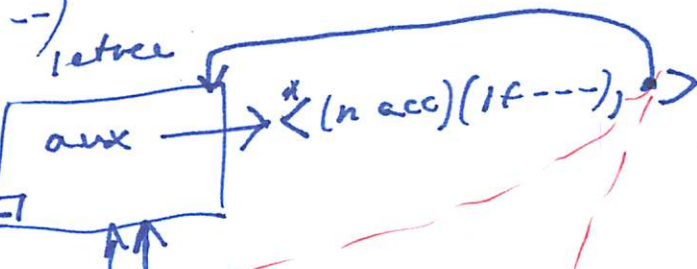
global



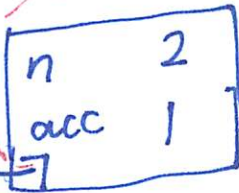
(factf 2)



(letrec ---) letrec

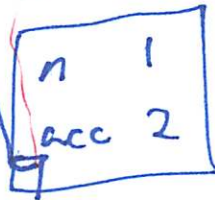


(aux n 1)
(-> 2 1)



(if (< n 2)

(aux 1 2)



(if (< n 2)

acc => 2
(aux -))

(define factf

(lambda (n)

(letrec

([aux (lambda (n acc)

(if (< n 2)

acc

(aux (- n 1)

(* n acc))))]

(aux n 1)))

(T)

```

(define cadd
  (lambda (x)
    (lambda (y)
      (+ x y))))

```

```

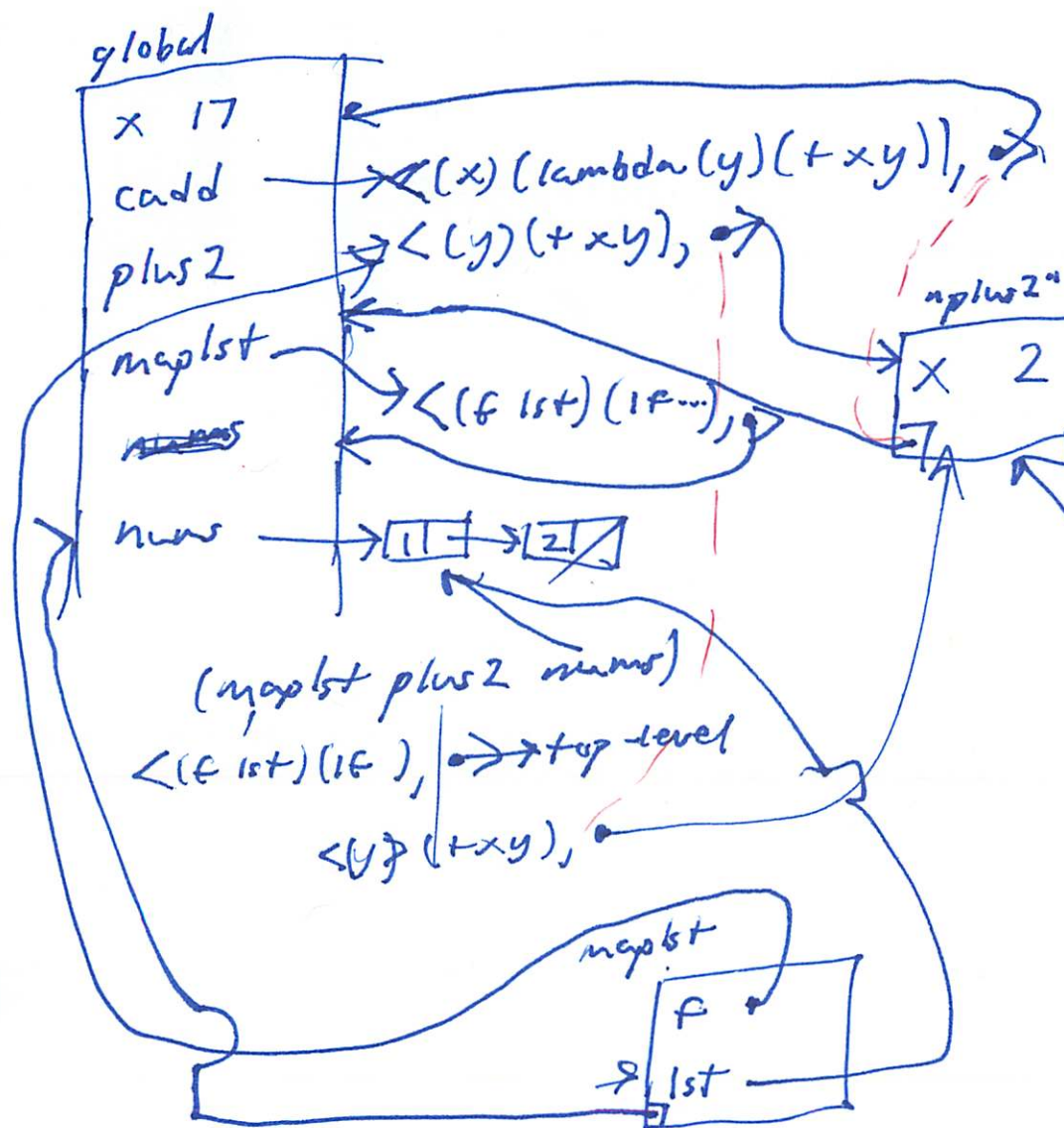
(define plus2
  (cadd 2))

```

```

(define maplst
  (lambda (f lst)
    (if (null? lst)
        '()
        (cons (f (car lst))
              (maplst f (cdr lst))))))

```



(maplst plus2 nums)
 \rightarrow top level
 $\langle (y) (+ x y) \rangle$

(if (null? lst) '())
 (cons (f 1) (maplst ---)))
 copy of closure for program
 $\langle (y) (+ x y) \rangle, 1$
 $(+ x y) \Rightarrow 3$

