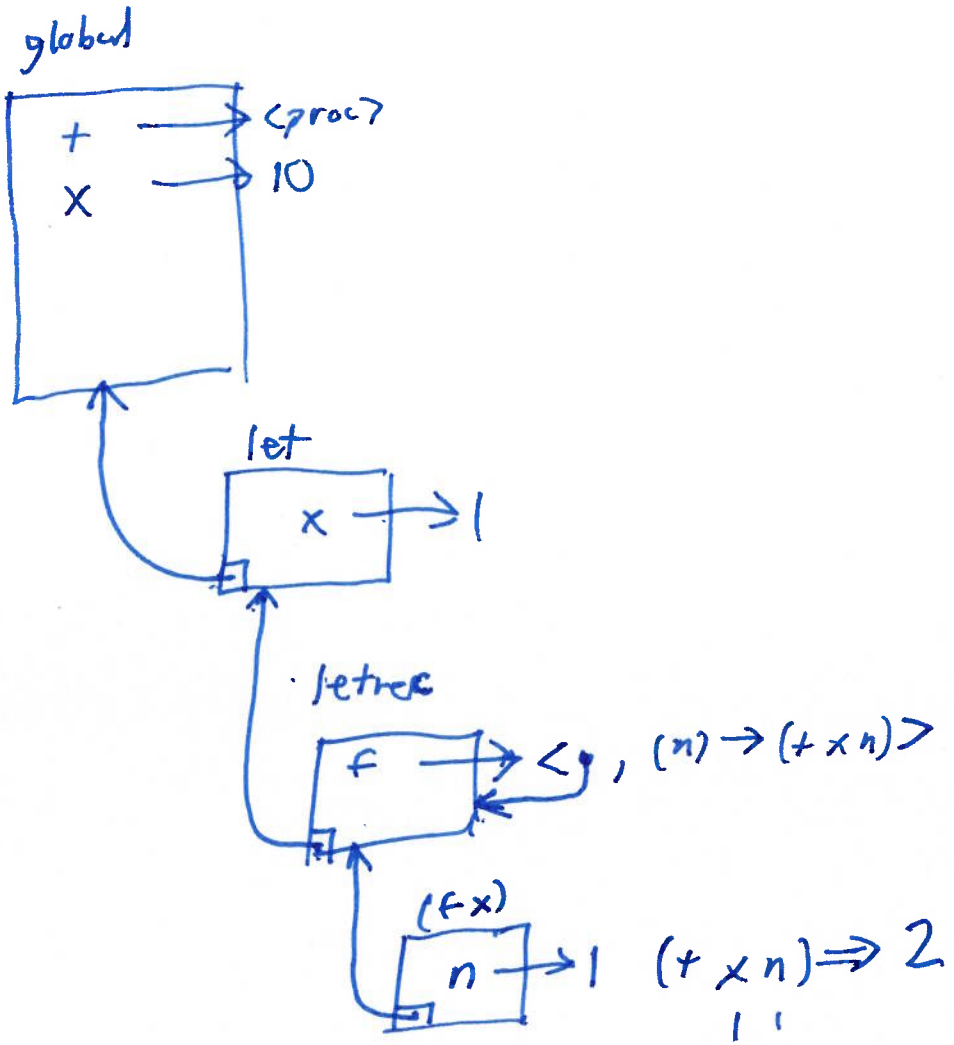
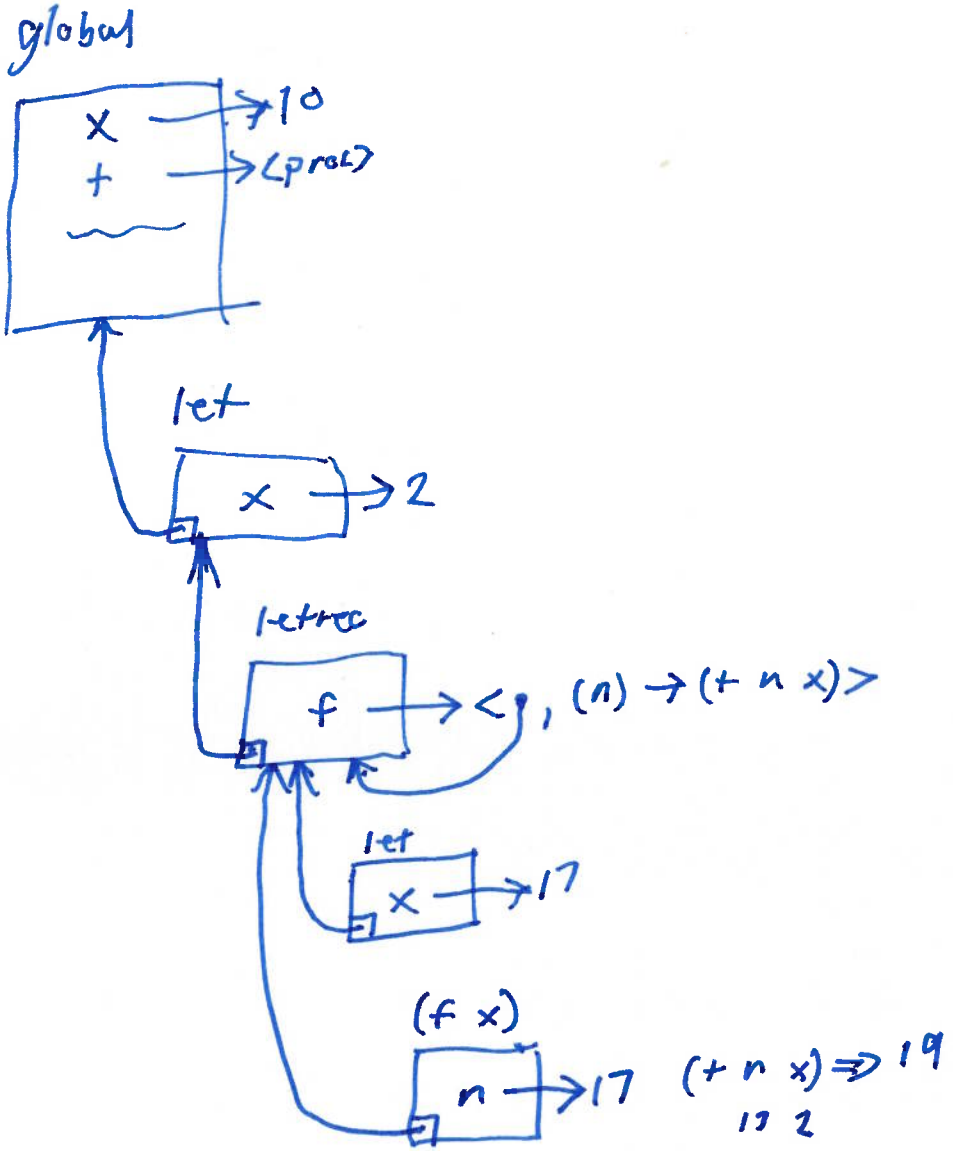
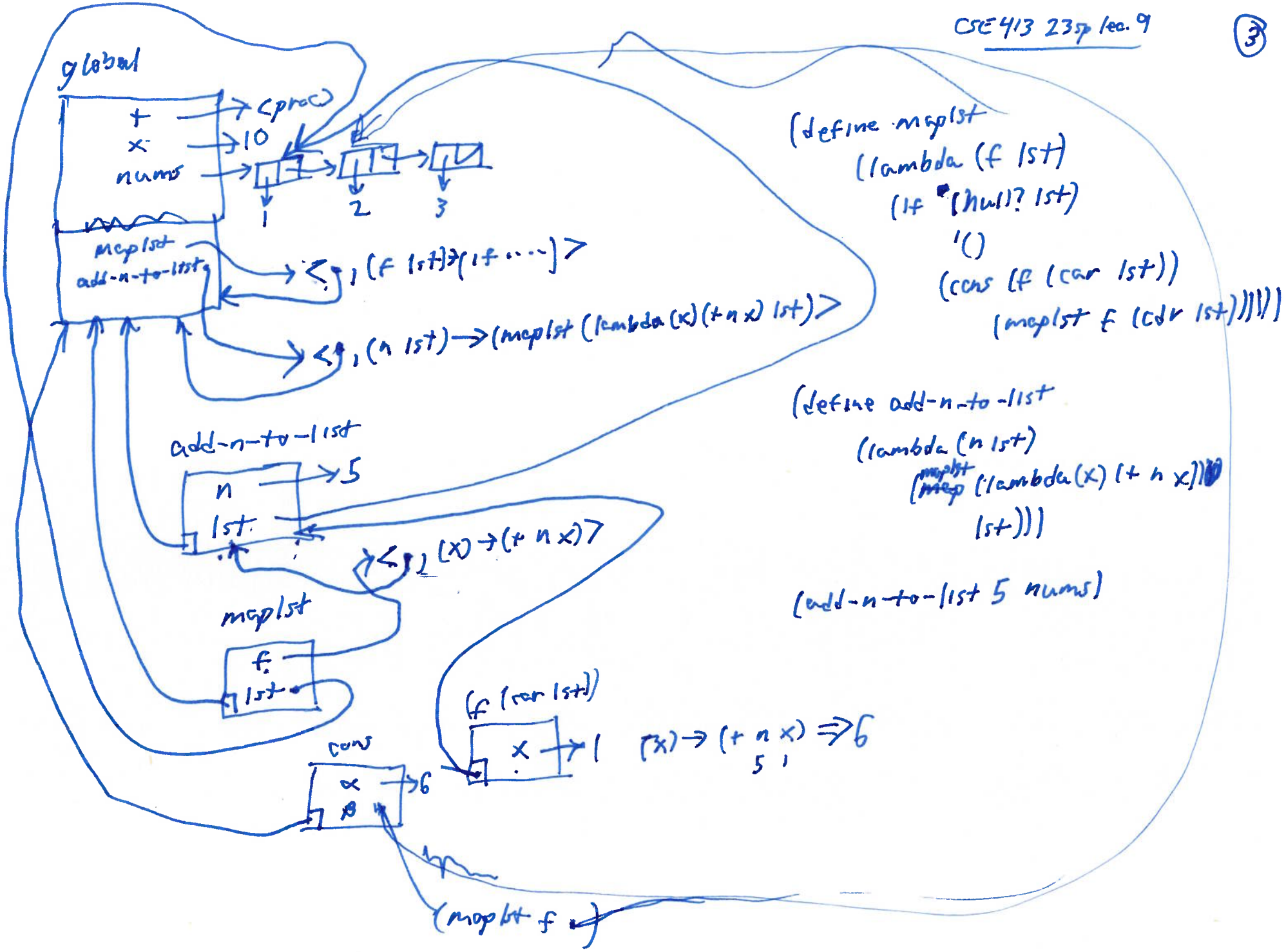
```
(let ([x 1])
  (letrec ([f (lambda (n) (+ x n))])
    (f x)))
```



```
(let ([x 2])
  (letrec ([f (lambda (n) (+ x n))])
    (let ([x 17])
      (f x))))
```




global
+
x
nums
mcplst
add-n-to-lst
cproc
10
1
2
3
<1, (f lst)→(if ...)>
<2, (n lst)→(mcplst (lambda (x)(+ n x)) lst)>
add-n-to-lst
n
lst
5
<1, (x)→(+ n x)>
mcplst
f
lst
(f (car lst))
x
1
(x)→(+ n x)
5
⇒6
cons
a
b
6
(mcplst f ...)
(define mcplst
(lambda (f lst)
(if (null? lst)
'()
(cons (f (car lst))
(mcplst f (cdr lst))))))
(define add-n-to-lst
(lambda (n lst)
(mcplst (lambda (x) (+ n x))
lst))))
(add-n-to-lst 5 nums)