# Clicker Questions
# for
# February 18

; map & apply

```
(define summer (lambda (L)
        (cond
                [(null? L) 0]
                [(else (apply + (map summer L)))])))
```

What is  (summer '( (1 2 3) (4 5 6)))?

A. 21  (=1+2+3+4+5+6))
B. (6 15)
C. 0
D. It causes an error

Answer D: (summer '( (1 2 3) (4 5 6))) generates an error

```
(define summer (lambda (L)
        (cond
                [(null? L) 0]
                [(else apply + (map summer L))])))
```

If you give summer a flat list such as (1 2 3) it tries to map summer onto the list, computing ( (summer 1)  (summer 2)  (summer 3) )  and those calls to summer crash.

; unrestricted lambda

I am trying to write (sumsq x y z w ...) that adds the squares of its arguments.

```
(define sumsq (lambda args
        (cond
                [(null? args) 0]
                [else (+ (* (car args) (car args)) (sumsq ???)])))
```

What goes in place of (sumsq ???)?  How does sumsq recurse on (cdr args)?

A.  (sumsq (cdr args))
B.  (apply sumsq (cdr args))
C.  (map sumsq (cdr args))
D. (cadr args)

Answer B: (apply sumsq (cdr args))