Clicker Questions
February 27

The syllabus says we will have an exam next Thursday (March 5). Does that seem like a good idea?

A. That's a splendid idea.
B. I would much rather have it on Tuesday, March 10.
C. Let's have exams both days! I love exams!
D. Personally, I'd rather not have an exam at all ....
E. Exam? We have exams in this class?????

How would you write (count a lat), which counts the number of times atom a appears in lat, with fold?

A. (define count (lambda (a lat)
(foldl (lambda (x y) (if (eq? x a) (+ y 1) y)) 0 lat)))
B. (define count (lambda (a lat)
(foldr (lambda (x y) (if (eq? x a) (+ y 1) x)) 0 lat)))
C. (define count (lambda (a lat)
(foldr (lambda (x y) (if (eq? x a) x y)) 0 lat)))
D. (define count (lambda (a lat)
(foldr (lambda (x y) (if (eq? x y) (+ y 1) y)) a lat)))

A. Answer A:

```
(define count (lambda (a lat)
        (foldl (lambda (x y) (if (eq? x a) (+ y 1) y)) 0 lat)))
```

I want to write a function sum2dVectors that adds the first elements of a bunch of pairs and then adds the second elements, so (sum2dVectors '(3 4) '(1 2) '(2 3)) returns '(6 9). The start is easy:

```
(define sum2dVectors (lambda pairs
        (cond
                [(null? (cdr pairs)) (car pairs)]
                [else (let ([a (car (car pairs))]
                            [b (cadr (car pairs))]
                            [v ;  THIS SHOULD BE THE RESULT OF
                                  sum2dVectors RECURSING ON ALL BUT ITS
                                  FIRST ARGUMENT
                    (list (+ a (car v)) (+ b (cadr v))))]))))
```

In the definition

```
(define sum2dVectors (lambda pairs ...
```

how does sum2dVectors recurse on all but its first argument

A. (sum2dVectors (cdr (list pairs)))
B. (sum2dVectors (cdr pairs))
C. (apply sum2dVectors (cdr pairs))
D. (apply (sum2dVectors (cdr pairs)))

Answer C: (apply sum2dVectors (cdr pairs))

How would you write sum2dVectors with foldr??

A.I wouldn't.

B.(define sum2dVectors (lambda pairs
(foldr (lambda (x y) (list (+ (car x) (car y)) (+ (cadr x) (cadr y))))
(list 0 0) pairs)))

C. (define sum2dVectors (lambda pairs
(foldr (lambda (x y) (+ x y)) (list 0 0) pairs)))

D. (define sum2dVectors (lambda pairs
(foldr (lambda (x y) (apply + x y)) (list 0 0) pairs)))

Answer B:

A.(define sum2dVectors (lambda pairs

(foldr (lambda (x y) (list (+ (car x) (car y)) (+ (cadr x) (cadr y))))

(list 0 0) pairs)))

How would you write sum2dVectors with map and apply??

A.I wouldn't.

B.(define sum2dVectors (lambda pairs
        (list (apply + (map car pairs)) (apply + (map cadr pairs)))

C. (define sum2dVectors (lambda pairs
        (apply + (map list pairs))

D. (define sum2dVectors (lambda pairs
        (apply list (map + pairs))

Answer B:

```
(define sum2dVectors (lambda pairs
        (list (apply + (map car pairs)) (apply + (map cadr pairs)))
```

Here's a way to do that in general

```
(define sumVectors (lambda vecs
        (map (lambda (p) (apply + p)) (apply map list vecs))))
```

For example (sumVectors '(1 2 3) '(4 5 6) '(7 8 9)) returns (12 15 18), (sumVectors '(1 2) '(3 4)) is (4 6) and so forth.