Suppose we make a 2D vector datatype with function

\[
\text{(define makeVector (lambda (a b) (list 'vector a b)))}
\]

How do we get the x-coordinate of a vector -- the a in \( \text{(makeVector a b)} \)?

A. \( \text{(define x-coord (lambda (vec) (car vec)))} \)

B. \( \text{(define x-coord (lambda (vec) (cdr vec)))} \)

C. \( \text{(define x-coord (lambda (vec) (cadr vec)))} \)

D. \( \text{(define x-coord (lambda (vec) (caddr vec)))} \)
Answer C: (define x-coord (lambda (vec) (cadr vec)))
Suppose we make a 2D vector datatype with function

\[
\text{(define makeVector (lambda (a b) (list 'vector a b)))}
\]

\[
\text{(define x-coord (vec) (cadr vec))}
\]

\[
\text{(define y-coord (vec) (caddr vec))}
\]

How could we add 2 vectors?
A. \text{(define add (lambda (v1 v2) (apply + v1 v2))}

B. \text{(define add ((+ (x-coord v1) (x-coord v2))(+ (y-coord v1) (y-coord v2))))}

C. \text{(define add (makeVector (map + (cdr v1) (cdr v2))))}

D. \text{(define add (makeVector (+ (x-coord v1) (x-coord v2))}

\[
(+ (y-coord v1) (y-coord v2))))
\]
A.  (define add (makeVector (+ (x-coord v1) (x-coord v2))
                   (+ (y-coord v1) (y-coord v2)))))
Suppose we make a 2D vector datatype with function

```
(define makeVector (lambda (a b) (list 'vector a b)))
```

How can we tell if something is a vector of this type?

A. `(define Vector? (lambda (x) (eq? 'vector x)))`
B. `(define Vector? (lambda (x) (eq? 'vector (car x))))`
C. `(define Vector? (lambda (x) (cond
       [(pair? x) (eq? 'vector (car x))]
       [else #f])))`
D. `(define Vector? (lambda (x) (and (number? (cadr x))
         (number? (caddr x)))))`
A. Answer C: (define Vector? (lambda (x)
   (cond
       [(pair? x) (eq? 'vector (car x))]
       [else #f])))

We are going to need a datatype for environments. Which Scheme expressions extend the environment?

A. let
B. let and lambda
C. let and lambda and calls
D. None of the above are completely correct
Answer D: None of the above are completely correct

The correct answer is let and call