fun returns lambda

What is the result of the following program?

```
(deffun (f x)
  (lambda (y) (+ x y)))

((f 2) 1)
```

- ○ 3
- ○ Error
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

filter gt

What is the result of the following program?

```
(filter (lambda (n) (> 3 n)) '(1 2 3 4 5))
```

- ‘(4 5)
- ‘(1 2)
- Other

Please specify

Why do you think this program was included? What do you think it's checking?

fun and state 1/4

What is the result of the following program?

```
(defvar x 1)
(defvar f
  (lambda (y)
    (+ x y)))
(set! x 2)
(f x)
```

- 4
- 3
- Other

Please specify

Why do you think this program was included? What do you think it's checking?

fun and state 2/4

What is the result of the following program?

```
(defvar x 1)
(deffun (f y)
  (+ x y))
(set! x 2)
(f x)
```

- ○ 4
- ○ 3
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

fun and state 3/4

What is the result of the following program?

```
(defvar x 1)
(defvar f
  (lambda (y)
    (+ x y)))
(let ([x 2])
  (f x))
```

- ○ 4
- ○ 3
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

fun and state 4/4

What is the result of the following program?

```
(defvar x 1)
(deffun (f y)
  (+ x y))
(let ([x 2])
  (f x))
```

- ○ 3
- ○ 4
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

eval order

What is the result of the following program?

```
(deffun (f x) (+ x 1))
(deffun (new-f x) (* x x))

(f (begin
     (set! f new-f)
     10))
```

- ○ 100
- ○ Error
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

Counter

What is the result of the following program?

```
(deffun (make-counter)
  (let ([count 0])
    (lambda ()
      (begin
        (set! count (+ count 1))
        count))))
(defvar f (make-counter))
(defvar g (make-counter))


(f)
(g)
(f)
(f)
(g)
```

- ○ 1; 1; 2; 3; 4
- ○ 1; 1; 2; 3; 2
- ○ 1; 1; 1; 1; 1
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

hof + set!

What is the result of the following program?

```
(defvar y 3)
(+ ((lambda (x) (set! y 0) (+ x y)) 1)
   y)
```

- ○ 4
- ○ 7
- ○ Error
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

filter

What is the result of the following program?

```
(defvar l (list (ivec) (ivec 1) (ivec 2 3)))
(filter (lambda (x) (vlen x)) l)
```

- ○ Error
- ○ '(#(1) #(2 3))
- ○ ‘(0 1 2)
- ○ '(#() #(1) #(2 3))
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

eq? fun fun 1/3

What is the result of the following program?

```
(eq? (λ (x) (+ x x))
     (λ (x) (+ x x)))
```

- ○ #t
- ○ #f
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?

eq? fun fun 2/3

What is the result of the following program?

```
(deffun (f x) (+ x x))
(deffun (g x) (+ x x))
(eq? f g)
```

- ◯ #f
- ◯ #t
- ◯ Other

Please specify

Why do you think this program was included? What do you think it's checking?

eq? fun fun 3/3

What is the result of the following program?

```
(deffun (f x) (+ x x))
(deffun (g) f)
(eq? f (g))
```

- ◯ #f
- ◯ #t
- ◯ Other

Please specify

Why do you think this program was included? What do you think it's checking?

equal? fun fun

What is the result of the following program?

```
(deffun (f) (lambda () 1))
(equal? (f) (f))
```

- ○ #t
- ○ #f
- ○ Other

Please specify

Why do you think this program was included? What do you think it's checking?