(Task 1 of 7) In this tutorial, we will review previously learned content. 0

Lispy |

(Task 2 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(defvar a 1)
(deffun (foo)
  (lambda (b)
    (+ a b)))
(defvar bar (foo))
(set! a 3)
(bar 0)
```

Pseudo

```
let a = 1
fun foo():
  return lam (b):
    return a + b
  end
end
let bar = foo()
a = 3
print(bar(0))
```

```
3
```

2

You predicted the output correctly 🎉🎉🎉 3

`a` is bound to `1`. `bar` is bound to the lambda function. `(set! a 3)` binds `a` to `3`. So, the value of `(bar 0)` is the value of `(+ 3 0)`, which is `3`.

Click here to run this program in the Stacker.

Lispy |

(Task 3 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(defvar a (mvec 88))
(defvar c (mvec a 88 88))
(set! a (mvec 76))
c
```

Scala 3

```
var a = Buffer(88)
var c = Buffer(a, 88, 88)
a = Buffer(76)
println(c)
```

```
#(#(88) 88 88)
```

5

You predicted the output correctly 🎉🎉🎉 6

`a` is first bound to a one-element vector. `c` is bound to a three-element vector, of which the first element is the one-element vector. `(set! a (mvec 76))` binds `a` to a new vector. This doesn't impact `c` because the 0-th element of `c` is still the one-element vector.

Click here to run this program in the Stacker.

Lispy |

(Task 4 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(defvar t 6)
(deffun (f1)
```

Scala 3

```
var t = 6
def f1 =
  t)
(deffun (f2)
  (set! t 4)
  (f1))
(f2)
(set! t 2)
(f1)
```

```
  t
def f2 =
  t = 4
  f1
println(f2)
t = 2
println(f1)
```

8

```
4 2
```

9

You predicted the output correctly 🎉🎉🎉

There is exactly one variable `t`. The `t` in `(set! t 4)` refers to that variable. Calling `f2` evaluates `(set! t 4)`, which mutates `t`. When the value of `t` is eventually printed, we see the new value.

Click here to run this program in the Stacker.

Lispy |

10

(Task 5 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(deffun (k b)
  (lambda (a)
    (+ a b)))
(defvar foo (k 3))
(defvar bar (k 2))
(foo 3)
(bar 3)
```

Python

```
def k(b):
    return lambda a: a + b
foo = k(3)
bar = k(2)
print(foo(3))
print(bar(3))
```

11

```
6 5
```

12

You predicted the output correctly 🎉🎉🎉

The value of `(k 3)` is a lambda function defined in an environment where `b` is bound to `3`. The value of `(k 2)` is *another* lambda function defined in an environment where `b` is bound to `2`. The two lambda functions are *different* values. So, the value of `(foo 3)` is `6`, while the value of `(bar 3)` is `5`.

Click here to run this program in the Stacker.

Lispy |

13

(Task 6 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(defvar n 2)
(deffun (h m)
  (set! m 7)
  n)
(h n)
```

JavaScript

```
let n = 2;
function h(m) {
  m = 7;
  return n;
}
console.log(h(n));
```

14

```
2
```

14

You predicted the output correctly 🎉🎉🎉 15

The function call binds `m` to `2`. The `set!` mutates the value of `m` to `7`, but `n` is still bound to `2`.

Click here to run this program in the Stacker.

Lispy | 16

(Task 7 of 7) What is the result of running this program?

Lispy [Run ▶]

```
(defvar n 3)
(defvar m n)
(set! n 6)
n
m
```

JavaScript

```
let n = 3;
let m = n;
n = 6;
console.log(n);
console.log(m);
```

```
6 3
```

17

You predicted the output correctly 🎉🎉🎉 18

The first definition binds `n` to `3`. The second definition binds `m` to the value of `n`, which is `3`. The `set!` mutates the binding of `n`, so `n` is now bound to `6`. But `m` is still bound to `3`.

Click here to run this program in the Stacker.

You have finished this tutorial 🎉🎉🎉

Please print the finished tutorial to a PDF file so you can review the content in the future. **Your instructor (if any) might require you to submit the PDF.**

Start time: 1738371737934