(Task 1 of 8) In this tutorial, we will review previously learned content. 0

Lispy |

(Task 2 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar n (* m 4))
(defvar m 3)

n
m
```

Pseudo

```
let n = m * 4
let m = 3
print(n)
print(m)
```

```
error
```
2

You predicted the output correctly 🎉🎉🎉 3

The first definition tries to bind `n` to the value of `(* m 4)`. To evaluate `(* m 4)`, we need the value of `m`. But `m` is not bound to a value at that moment.

Click here to run this program in the Stacker.

Lispy |

(Task 3 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(deffun (foobar n)
  (defvar m 4)
  (+ n m))

(+ (foobar 2) m)
```

Python

```
def foobar(n):
    m = 4
    return n + m
print(foobar(2) + m)
```

```
error
```
5

You predicted the output correctly 🎉🎉🎉 6

`(+ (foobar 2) m)` is evaluated in the top-level block, where `m` is not defined. So, this program errors.

Click here to run this program in the Stacker.

Lispy |

(Task 4 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar m1 (mvec 77 77))
(deffun (f m2)
  (vec-set! m2 0 43))
(f m1)
m1
```

Scala 3

```
var m1 = Buffer(77, 77)
def f(m2 : Buffer[Int]) =
  m2(0) = 43
f(m1)
println(m1)
```

8

```
#(43 77)
```

You predicted the output correctly 🎉🎉🎉 9

The program first creates a two-element vector. The elements are both `77`. After that, the program defines a function `f`. The function call `(f m1)` replaces the first vector element with `43`. After that, the vector is printed. The vector now refers `43` and `77`, so the result is `#(43 77)`.

Click here to run this program in the Stacker.

Lispy |

(Task 5 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar v1 (mvec 23))
(defvar v2 v1)
(vec-set! v1 0 45)
v2
```

Pseudo

```
let v1 = vec[23]
let v2 = v1
v1[0] = 45
print(v2)
```

```
#(45)
```
11

You predicted the output correctly 🎉🎉🎉 12

The program creates a one-element vector (the only element being 23) and binds it to both `v1` and `v2`. The `0`-th element of the vector is then replaced with `45`. So, when the vector is printed as `#(45)`.

Click here to run this program in the Stacker.

Lispy |

(Task 6 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar v1 (mvec 53))
(defvar v2 (mvec 72 v1))
(vec-set! v1 0 72)
v2
```

Pseudo

```
let v1 = vec[53]
let v2 = vec[72, v1]
v1[0] = 72
print(v2)
```

```
#(72 #(72))
```
14

You predicted the output correctly 🎉🎉🎉 15

`v1` is bound to a one-element vector. The only element of this vector is `53`. `v2` is bound to a two-element vector. The elements of this vector are `72` and the one-element vector. The subsequent `(vec-set! v1 0 72)` mutates the one-element vector. Finally, the value of `v2` is printed.

Click here to run this program in the Stacker.

Lispy |

(Task 7 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar n 5)
(deffun (f1 m)
  (deffun (f2)
    (defvar l 4)
    (+ n m l))
  (f2))
(+ (f1 1) 3)
```

Python

```
n = 5
def f1(m):
    def f2():
        l = 4
        return n + m + l
    return f2()
print(f1(1) + 3)
```

17

```
13
```

18

You predicted the output correctly 🎉🎉🎉

This program binds `n` to `5` and `f1` to a function, and then evaluates `(+ (f1 1) 3)`. The value of `(+ (f1 1) 3)` is the value of `(+ (f2) 3)`, which is the value of `(+ (+ n m l) 3)`, which is the value of `(+ (+ 5 1 4) 3)`, which is `13`.

Click here to run this program in the Stacker.

Lispy | 🐞

(Task 8 of 8) What is the result of running this program?

Lispy [Run ▶]

```
(defvar m (mvec 82 76))
(vec-set! m 0 m)
(vec-ref m 1)
```

JavaScript

```
let m = [ 82, 76 ];
m[0] = m;
console.log(m[1]);
```

20

```
76
```

21

You predicted the output correctly 🎉🎉🎉

`m` is bound to a vector. `(vec-set! m 0 m)` replaces the 0-th element of the vector with the vector itself. This is fine because a vector element can be any value, including itself. Besides, the vector is not copied, so the mutation finishes immediately.

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: 1738524594523