Lispy |

(Task 1 of 6) In this tutorial, we will learn about **mutable values**, illustrated with **vectors**.

The following program illustrates how to create vectors

Lispy [Run] Python

```
(mvec (mvec 1 2) (mvec 3) (mvec))   print([ [ 1, 2 ], [ 3 ], [  ] ])
```

This program produces `#(#(1 2) #(3) #())`. It creates four vectors:

- a two-**element** vector that refers `1` and `2`
- a one-element vector that refers `3`
- an empty vector
- a three-element vector that refers all three aforementioned vectors

Lispy |

(Task 2 of 6) What is the result of running this program?

Lispy [Run] Python

```
(defvar n 44)     n = 44
(deffun (f x)     def f(x):
  (+ x 1))            return x + 1
(mvec n (f n))    print([ n, f(n) ])
```

`#(44 45)`

2

3

You predicted the output correctly 🎉🎉🎉

This program binds `n` to `44` and `f` to a function that adds 1 to its parameter. After that, the program creates a vector that refers the value of `n`, which is `44`, and the value of `(f n)`, which is `45`. The vector is printed as `#(44 45)`.

Click here to run this program in the Stacker.

Lispy |

(Task 3 of 6) The following program illustrates how to refer to vector elements.

Lispy [Run] Scala 3

```
(defvar v (mvec 84 75))   val v = (84, 75)
(vec-ref v 0)             println(v(0))
(vec-ref v 1)             println(v(1))
(vec-ref v 2)             println(v(2))
```

This program produces `84 75 error`. It refers to the `0`-th (i.e., first) element, the `1`-th element, and then tries to refer to the `2`-th element.

Lispy |

(Task 4 of 6) What is the result of running this program?

Lispy [Run]

```
(defvar v (mvec (mvec 58 43) (mvec 43 66)))
(defvar vr (vec-ref v 1))
(vec-ref vr 0)
```

Scala 3

```
val v = ((58, 43), (43, 66))
val vr = v(1)
println(vr(0))
```

6

```
43
```

7

You predicted the output correctly 🎉🎉🎉

Recall that the left-most element is the 0-th (rather than 1-th!) element. `(vec-ref v 1)` produces the value of `(mvec 43 66)`. So, `(vec-ref vr 0)` produces `43`.

Click here to run this program in the Stacker.

Lispy |

(Task 5 of 6) The following program illustrates how to mutate vectors.

Lispy [Run]

```
(defvar m (mvec 62 77))
(vec-set! m 0 83)
(vec-ref m 0)
```

Pseudo

```
let m = vec[62, 77]
m[0] = 83
print(m[0])
```

This program produces `83`. It **mutates** the vector by **replacing** the `0`-th element with `83` and then refers to the `0`-th element.

Lispy |

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

Lispy [Run]

```
(defvar x (mvec 92 73))
(vec-set! x 0 67)
x
```

JavaScript

```
let x = [ 92, 73 ];
x[0] = 67;
console.log(x);
```

10

```
67 73
```

11

The answer is `#(67 73)`.

▼ Textual explanation

`x` is bound to a vector. A vector referring `67` and `73` is printed as `#(67 73)`.

Click here to run this program in the Stacker.

Lispy |

What is the result of running this program?

Lispy [Run]

Python

```
(defvar v (mvec 83 64))
```

```
v = [ 83, 64 ]
```

```
(deval v (mvec 83 64))   v = [ 83, 64 ]
(vec-set! v 0 66)        v[0] = 66
v                        print(v)
```

```
#(66 64)
```

13

You predicted the output correctly 🎉🎉🎉

14

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: 1737146174687