CS 61A Spring 2021

HOFs, Lambdas, Self-Reference

Lost 2: February 5, 2021 Solutions

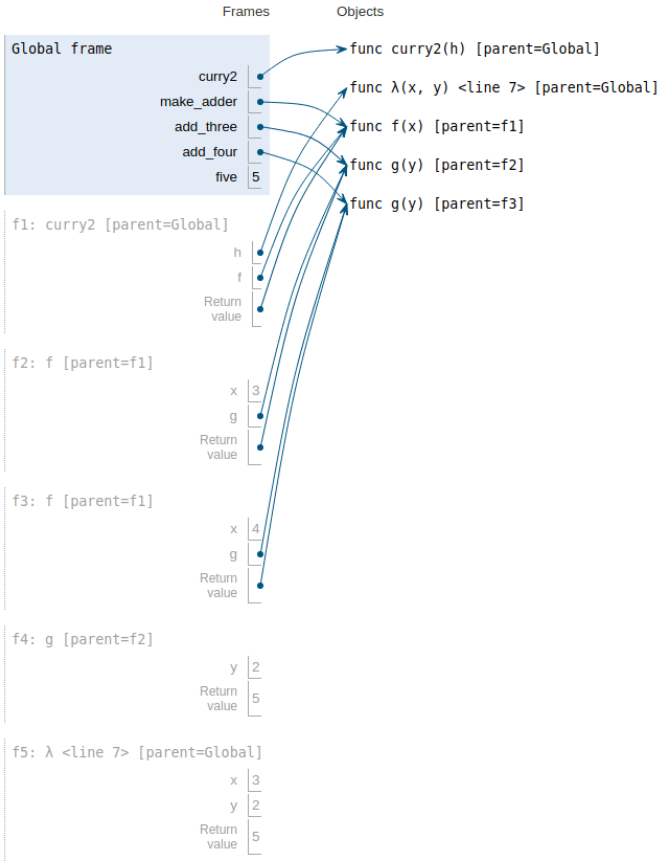
1 Learning Goals

- Review the basics of environment diagrams and understand them at a deeper level
- Understand the idea of function self-reference as a prelude to recursion (which we will do next week)
- Review higher-order functions and lambda functions at a high level
- Learn how to approach more challenging higher-order function and lambda problems

2 Orientation/Tutorial Review

2.1 Draw the environment diagram that results from executing the code below.

```
1  def curry2(h):
2      def f(x):
3          def g(y):
4              return h(x, y)
5          return g
6      return f
7  make_adder = curry2(lambda x, y: x + y)
8  add_three = make_adder(3)
9  add_four = make_adder(4)
10 five = add_three(2)
```



Solution: pythontutor

2.2 Write `curry2` as a lambda function.

```
curry2 = lambda h: lambda x: lambda y: h(x, y)
```

Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.

2.3 Write a function `print_delayed` that delays printing its argument until the next function call. `print_delayed` takes in an argument `x` and returns a new function `delay_print`. When `delay_print` is called, it prints out `x` and returns another `delay_print`.

```
def print_delayed(x):
    """Return a new function. This new function, when called,
    will print out x and return another function with the same
    behavior.
    >>> f = print_delayed(1)
    >>> f = f(2)
    1
    >>> f = f(3)
    2
    >>> f = f(4)(5)
    3
    4
    >>> f("hi")
    5
    <function print_delayed> # a function is returned
    """
    def delay_print(y):
        ____________________________
        return _____________________
    return delay_print
```

```
def print_delayed(x):
    def delay_print(y):
        print(x)
        return print_delayed(y)
    return delay_print
```

3 Additional Practice (Medium-Level Difficulty)

3.1 The following code has been loaded into the Python interpreter:

```
def skipped(f):
    def g():
        return f
    return g
def composed(f, g):
    def h(x):
        return f(g(x))
    return h
def added(f, g):
    def h(x):
        return f(x) + g(x)
    return h
def square(x):
    return x*x
def two(x):
    return 2
```

What will Python output when the following lines are evaluated?

```
>>> composed(square, two)(7)
```

4

```
>>> skipped(added(square, two))()(3)
```

11

```
>>> composed(two, square)(2)
```

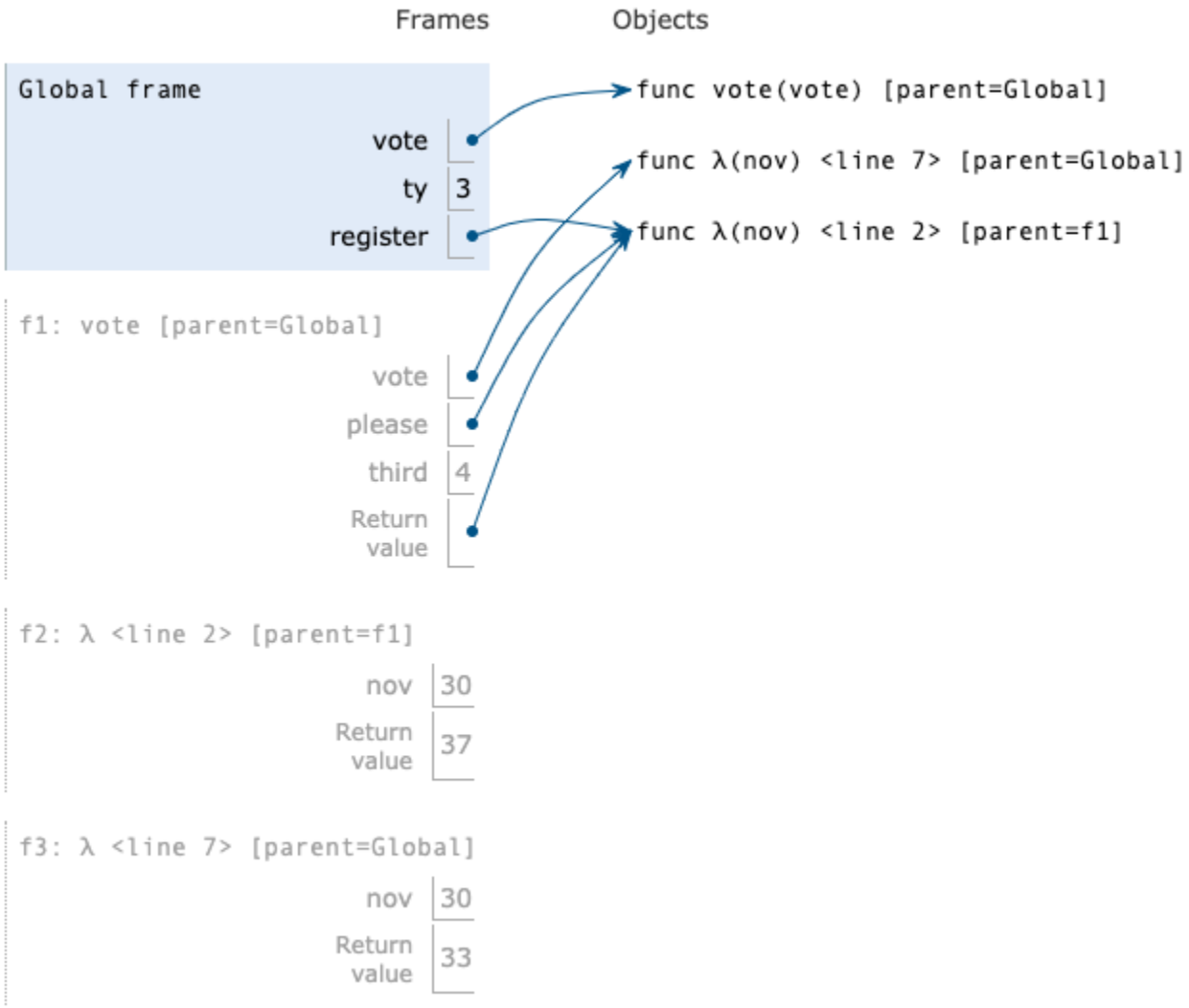
2

4 Exam-Level Practice

4.1 **Fall 2020 Midterm 1, Question 3** Fill in each example in the code example below so that its environment diagram is what you see on the following page:

```
def vote(vote):
    please = ______
    ______ = ty + 3
    return please
ty = 1
register = ______(lambda nov: nov + ty)
______
register(_____)
```

```
def vote(vote):
    please = lambda nov: vote(nov) + third
    third = ty + 3
    return please
ty = 1
register = vote(lambda nov: nov + ty)
ty = 3
register(ty * 10)
```



Note: This worksheet is a problem bank—most TAs will not cover all the problems in discussion section.