Started on	Wednesday, 20 April 2022, 10:50 AM
State	Finished
Completed on	Wednesday, 20 April 2022, 10:57 AM
Time taken	7 mins 49 secs
Marks	1.50/2.00
Grade	**7.50** out of 10.00 (**75**%)

Question 1

Correct

Mark 1.00 out of 1.00

True or False? In simply typed lambda calculus with imperative objects, a fix point combinator is used during the construction of an object to allow object methods to refer to each other.

Select one:

- True ✔
- False

The correct answer is 'True'.

Question **2**

Partially correct

Mark 0.50 out of 1.00

What does the following program in simply typed lambda calculus with imperative objects evaluate to?

Type alias declarations:

```
Counter    = { get : Unit → Nat, inc : Unit → Unit }
SetCounter = { set : Nat → Nat, get : Unit → Nat, inc : Unit → Unit }
LogCounter = { set : Nat → Nat, get : Unit → Nat, inc : Unit → Unit, logCount : Unit → Nat }

CounterRep = { x : Ref Nat }
LogCounterRep = { x : Ref Nat, log : SetCounter }
```

The program:

```
let setCounterClass : CounterRep → SetCounter → SetCounter =
  λrep:CounterRep.
    λthis:SetCounter.
      { get = λ_:Unit. rep.x
      , set = λn:Nat. rep.x := n
      , inc = λ_:Unit. this.set (succ (this.get unit))
      }
in let newSetCounter : Unit → SetCounter =
  λ_:Unit. let rep = { x = 0 } in fix (setCounterClass rep)
in let logCounterClass : LogCounterRep → LogCounter → LogCounter =
  λrep:LogCounterRep.
    λthis:LogCounter.
      let super = setCounterClass rep this in
        { get = λ_:Unit. rep.log.inc; super.get unit
        , set = super.set
        , inc = super.inc
        , logCount = rep.get unit
        }
in let newLogCounter : Unit → LogCounter =
   λ_:Unit. let rep = { x = 0, log = newSetCounter unit } in fix (logCounterClass rep)
in let test : Counter → Nat =
   λc:Counter. c.inc unit; c.inc unit; c.inc unit; c.get
in let c : LogCounter = newLogCounter unit
in c.set (test c); c.logCount
```

Answer: 3

The correct answer is: 4

◄ Quiz 7 — Apr 14 from 9:10 to 9:20 (10 minutes)

Jump to...

Quiz 9 — Apr 21 from 9:10 to 9:20 (10 minutes) ►