Concurrency Quiz: Solution SCOOP

First name, last name: ....

Background information

In this part of the quiz, we would like to collect some information concerning your prior experience with concurrent programming.

What level of studies are you currently completing?
☐ Bachelor in Computer Science
☐ Master in Computer Science
☐ PhD in Computer Science

☐ Other: ....

Which semester are you currently completing? (e.g. `4th`)................................

Prior experience with concurrency

Have you ever taken or are you currently taking a course other than Software Architecture that covers concurrent programming?
☐ Yes
☐ No
☐ No, but I studied it on my own (e.g. through online tutorials, books, ...)

If yes, what course was/is it and when did you take it? (Please provide details below.)

☐ Parallel programming @ ETH Zurich by T. Gross in Spring ....

☐ Concepts of concurrent computation @ ETH Zurich by B. Meyer in Spring ....

☐ Other courses:

....

....

How much of the self-study material on concurrency that you worked with today did you already know before?

☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
none	10%	20%	30%	40%	50%	60%	70%	80%	90%	all

Programming experience (sequential and concurrent)

	(1: a novice . . . 5: an expert)
Concerning your **general programming experience**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Concerning your experience with **concurrent programming**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Concerning your experience with the programming language **Eiffel**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Concerning your experience with the programming language **Java**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Concerning your experience with **Java Threads**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
Concerning your experience with **SCOOP**, do you consider yourself. . .	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Self-study material

Where did you work through the self-study material?

☐ In the morning lecture ☐ In the exercise class ☐ At home

	(1: totally disagree . . . 5: totally agree)
The self-study material was easy to follow.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
The self-study material provided enough **examples** to help me understand the subject.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
The self-study material provided enough **exercises** to help me understand the subject.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
I was able to complete the tutorial within 90 minutes.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
The self-study material is a good alternative to the traditional lectures.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
I feel confident that I will be able to solve the tasks in this quiz.	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

Any comments on the self-study material:. . . . .

. . . .

. . . .

. . . .

. . . .

1 Sequential comprehension

Write down the output of the sequential Eiffel program shown below.

AFSTML

```
class APPLICATION
create
  make

feature
  a: A
  b: B
  c: C

  make
    do
      create a; create b;
      create c;
      print ("A")
      run (a, b)
      print ("L")
    end

  run (aa: A; bb: B)
    do
      aa.m(c)
      bb.n(c)

      if (aa.done_a and bb
          .done_b)
      then
        print ("M")
      end
    end

end
```

```
class A

feature

  done_a: BOOLEAN

  m (cc: C)
    do
      cc.f
      done_a := true
    end

end
```

```
class B

feature

  done_b: BOOLEAN

  n(cc: C)
    do
      cc.g
      done_b := true
      print ("T")
    end

end
```

```
class C

feature

  done_c: BOOLEAN

  f
    do
      print("F")
      done_c := false
    end

  g
    do
      print("S")
      done_c := true
    end

end
```

2 General concurrency concepts

What is multiprocessing?

☐ Execution of multiple processes, within a single computer sharing a single processing unit.
☐ Execution of a single process on a single computer.
☐ Execution of a single process within multiple computers.
☐ Execution of multiple processes within a single computer sharing two or more processing units.

d is the correct answer

Which of the following state transitions is not possible in the status of a process?

☐ running → ready
☐ ready → blocked
☐ blocked → ready
☐ running → blocked

b is the correct answer

In the space below explain the terms data race and mutual exclusion.

See self-study material.
What is a deadlock?

See self-study material.

3 Comprehension

Write down three possible (non-deadlock) outputs for the SCOOP program shown below:

Some possible sequences: APTSFFTSML, ATSFPFTSML, ATSPFTSFML, APTTSSFFML, ATPTSFSFML, APTTSFFSML.

```
class APPLICATION
create
  make

feature
  a: separate A
  b: separate B
  c: separate C
  d: D

  make
    do
      create a; create b;
      create c; create d;
      print ("A")
      run (a, b)
      print ("L")
    end

  run (aa: separate A;
      bb: separate B)
    do
      aa.m (c)
      bb.n (c)
      aa.m (c)
      bb.n (c)
      d.foo

      if (aa.done_a and
          bb.done_b)
      then
        print ("M")
      end
    end
end
```

```
class A
feature
  done_a: BOOLEAN

  m (cc: separate C)
    require
      cc.done_c
    do
      cc.f
      done_a := true
    end
end
```

```
class B
feature
  done_b: BOOLEAN

  n(cc: separate C)
    require
      not cc.done_c
    do
      print ("T")
      cc.g
      done_b := true
    end
end
```

```
class C
feature
  done_c: BOOLEAN

  f
    do
      print("F")
      done_c := false
    end

  g
    do
      print("S")
      done_c := true
    end
end
```

```
class D
feature
  foo
    do
      print("P")
    end
end
```

4 Errors

Identify errors (possibly compile-time) in the following SCOOP code segment. Justify your answers by providing on the next page the line number and a short explanation for every detected error. (The number of provided spaces does not necessarily correspond to the actual number of errors.)

```
 1 class A create make
 2 feature
 3   b: separate B
 4   c: C
 5
 6   make
 7     local
 8       b1: B;
 9     do
10       create b; create c;
11       b1 := g
12     end
13
14   f(b1: separate B): B
15     local
16       b2: B
17       c1: separate C
18     do
19       b2 := b
20       c1 := c
21       b.f
22       c.g
23       Result := b1.h
24     end
25
26   g: separate B
27     local
28       b1: B
29     do
30       h (b)
31       create b1
32       Result := b1
33     end
34
35   h(b1: B)
36     local
37       b2: separate B
38       c1: C
39       i: INTEGER
40     do
41       create b2
42       i := c.r
43       c1 := b2.h.r
44     end
45 end
```

```
46 class B
47
48 feature
49
50   r: C
51
52   h: B
53
54   f
55     do
56       create h
57       create r
58     end
59 end
```

```
60 class C
61
62 feature
63
64   r: INTEGER
65
66   g
67     do
68       r := 10
69     end
70 end
```

Some of the errors that could be mentioned include:

- Line 11: assignment of *b1* := *g* is not correct as *g* returns a separate object.
- Line 19: assignment of *b2* := *b* is not correct as *b* is separate.
- Line 21: *b.f* : routine needs to be wrapped. Violates the Separate Call rule: The target of a separate call must be a formal argument of the routine in which the call appears.
- Line 23 or Line 14: since *b1* is separate, also *b1.h* is separate with respect to the current object. However the result type is non-separate, which violates the typing rules.
- Line 30 or Line 35: *h*(*b*) passes a separate entity as the actual parameter.
- Line 43: the right hand side of the assignment returns a separate entity, but *c1* is non-separate.

5 Program Construction

Consider a class *Data* with two integer fields *x* and *y*, both of which are initialized to 0. Two classes *C0* and *C1* share an object *data* of type *Data*. Class *C0* implements the following behavior, which is repeated continuously: if both values *data.x* and *data.y* are set to 1, it sets both values to 0; otherwise it waits until both values are 1. Conversely, class *C1* implements the following behavior, which is also repeated continuously: if both values *data.x* and *data.y* are set to 0, it sets both values to 1; otherwise it waits until both values are 0. The following condition must always hold when *data* is accessed:

$$(data.x = 0 \land data.y = 0) \lor (data.x = 1 \land data.y = 1)$$

Write a concurrent program using SCOOP that implements the described functionality. Besides the mentioned classes *Data*, *C0*, and *C1*, your program needs to have a root class which ensures that the behaviors of *C0* and *C1* are executed on different processors.

```
class
  APPLICATION

create
  make

feature
  data: separate DATA
  c0: separate C0
  c1: separate C1

  make
    do
      create data
      create c0.make (data)
      create c1.make (data)

      run (c0, c1)
    end

  run (cc0: separate C0; cc1: separate C1)
    do
      cc0.run
      cc1.run
    end

end

class DATA

feature
  x, y: INTEGER

  set_x (v: INTEGER)
```

```
        do
            x := v
        end

    set_y (v: INTEGER)
        do
            y := v
        end

end

class C0

create
    make

feature
    data: separate DATA

    make (d: separate DATA)
        do
            data := d
        end

    run
        do
            from until False
            loop
                set_0 (data)
            end
        end

    set_0 (d: separate DATA)
        require
            d.x = 1 and d.y = 1
        do
            d.set_x (0)
            d.set_y (0)
        end
end

class C1

create
    make

feature
    data: separate DATA
```

```
   make (d: separate DATA)
      do
         data := d
      end

   run
      do
         from until False
         loop
            set_1 (data)
         end
      end

   set_1 (d: separate DATA)
      require
         d.x = 0 and d.y = 0
      do
         d.set_x (1)
         d.set_y (1)
      end
end
```

Feedback on the quiz

How much time did you spend on this quiz?

20’	30’	40’	50’	60’	70’	80’	90’	100’	110’	120’
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

The difficulty level of the quiz was… (1: too easy, 2: easy, 3: just right, 4: difficult, 5: too difficult)	1 2 3 4 5 ☐ ☐ ☐ ☐ ☐
I feel confident that I solved the tasks of this quiz correctly. (1: totally disagree … 5: totally agree)	1 2 3 4 5 ☐ ☐ ☐ ☐ ☐

Did you leave any questions of the quiz empty and if so, why? ....

....

....

....

....

....

....

....

Any comments on the quiz: ....

....

....

....

....

....

....

....