Module 4 Quiz

Quiz, 10 questions

1 point

1.
Based on the simple code example in the Topic 4.2 lecture video, which of the following are true of phasers?

- A. Using phasers can help to reduce both the critical path and work of this parallel program.
- B. Using phasers can help to reduce the critical path but does not affect the work of this parallel program.
- C. Using phasers increases the work of this parallel program but does not affect the critical path.
- D. Using phasers increases both the critical path and work of this parallel program.

---

1 point

2.
True or False, if a given thread has hit a *next* and is now in wait mode, then we can state that at least some other threads have also hit a *next* and have done a signal of the phaser.

- True
- False

---

1 point

3.
Given three tasks and three phasers (ph0, ph1, and ph2), which of the following code snippets uses phasers to implement what is semantically a barrier?

- A.

Module 4 Quiz

Quiz, 10 questions

```
// Task 0
ph0.signal();
ph0.wait();

// Task 1
ph1.signal();
ph1.wait();

// Task 2
ph2.signal();
ph2.wait();
```

B.

```
// Task 0
ph0.signal();
ph0.wait();

// Task 1
ph1.signal();
ph0.wait();

// Task 2
ph2.signal();
ph0.wait();
```

C.

```
// Task 0
ph0.signal();
ph1.wait();

// Task 1
ph1.signal();
ph2.wait();

// Task 2
ph2.signal();
ph0.wait();
```

D.

Module 4 Quiz

Quiz, 10 questions

```
1  // Task 0
2  ph0.signal();
3  ph1.wait();
4  ph2.wait();
5
6  // Task 1
7  ph1.signal();
8  ph0.wait();
9  ph2.wait();
10
11 // Task 2
12 ph2.signal();
13 ph0.wait();
14 ph1.wait();
```

---

1 point

4.

What is the primary benefit of using a phaser over a barrier?

- A. Using a phaser rather than a barrier always reduces the critical path of a program.
- B. Phasers allow for a more precise definition of the dependencies in a parallel program, potentially exposing more parallelism.
- C. The use of phasers guarantees data race freedom in multi-threaded Java programs.
- D. The use of phasers guarantees deadlock freedom in multi-threaded Java programs.

---

1 point

5.

True or False, a child task waiting on a phaser registered with the parent task will cause a deadlock if the parent task reaches the end of the scope in which the phaser is declared without issuing a signal.

- True
- False

Module 4 Quiz

Quiz, 10 questions

1 point

6.

Consider the example pipeline in the Topic 4.4 lecture video. If instead of using phasers we used barriers to implement synchronization between the pipeline stages, would you expect performance to improve, worsen, or remain the same?

- Improve
- Stay the same
- Worsen

---

1 point

7.

Given a hardware platform with C cores, assuming an infinite supply of equally sized and immediately available inputs, assuming that each pipeline stage can only process one input at a time, and assuming that each pipeline stage takes the same amount of time, how would the speedup of a parallel pipeline scale with the number of pipeline stages P? Ignore any effects from warming up the pipeline.

- A. The speedup achieved by a parallel pipeline would scale linearly with P up until P == C, and then be limited to Cx speedup.
- B. The speedup would always equal C, regardless of number of stages.
- C. The speedup achieved by a parallel pipeline would scale linearly for all P.
- D. Because each stage can only process one input at a time, speedup would never go beyond 1x.

---

1 point

8.

True or False, the order in which dataflow asyncAwait tasks are launched affects the logical ordering of their execution.

- True
- False

Module 4 Quiz

Quiz, 10 questions

1 point

9.

Below are three code snippets, each containing the definition of a task from the same program. These tasks use three phasers (ph0, ph1, and ph2) to synchronize among themselves. Which of the code snippet options using asyncAwait is semantically equivalent to these three phaser-synchronized tasks?

```
1  async {
2    A();
3    ph0.signal();
4  }
5
```

```
1  async {
2    ph0.wait();
3    B();
4    ph1.signal();
5  }
6
```

```
1  async {
2    ph1.wait();
3    C();
4    ph2.signal();
5  }
6
```

A.

```
1  asyncAwait(f1) { C(); f2.put(); }
2  asyncAwait(f0) { B(); f1.put(); }
3  async { A(); f0.put(); }
```

B.

```
1  asyncAwait(f0) { A(); f0.put(); }
2  asyncAwait(f1) { B(); f1.put(); }
3  asyncAwait(f2) { C(); f2.put(); }
```

C.

```
1  async { f0.put(); A(); }
2  asyncAwait(f0) { f1.put(); B(); }
3  asyncAwait(f1) { f2.put(); C(); }
```

Module 4 Quiz

Quiz, 10 questions

D.

```
1  async { f0.put(); A(); }
2  asyncAwait(f0) { B(); f0.put(); }
3  asyncAwait(f0) { C(); f0.put(); }
```

1 point

10.

True or False, there are computation graphs that can be expressed using explicit waits on futures (e.g. A.get()) that cannot be expressed using dataflow programming (i.e. asyncAwait(A)).

- True
- False

I, **Fan Yang**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Submit Quiz