Module 2 Quiz

,

People are making progress

157 learners have recently completed this assignment

Submit your assignment
Due DateDec 21, 12:59 AM MST
Attempts3 every 8 hours
Receive grade
To Pass70% or higher
Grade
—

Module 2 Quiz
Graded Quiz • 30 min

Due Dec 21, 12:59 AM MST

Module 2 Quiz

Total points 10
1.
Question 1

Consider the following parallel pseudo-code that use future tasks, as introduced in Lecture 2.1. Select the statement below which must be true for this code snippet.

```
finish {
  future A = future { S1; return 1;} ;
  future B = future { S2; int a = A.get(); S3; return 3;} ;
  future C = future { int a = A.get(); S4; int b = B.get();
                  S5; return 5;} ;
}
```

1 point

A. S1 can run in parallel with S3.

B. S2 can run in parallel with S5.

C. S1, S2, and S4 can all run in parallel with each other

D. S2 can run in parallel with S4.

E. S4 can run in parallel with S5.

2.
Question 2

Which of the following dependencies result from the following pseudo-code using futures (as introduced in Lecture 2.1)?

```
finish {
  future A = future { int a = 1; S1; S2; return a;} ;
  future B = future { int b = 2; S3; int a = A.get(); S4;
                      return a+b;} ;
}
```

1 point

A. S3 depends on S2 having executed

B. S4 depends on S2 having executed

C. S2 depends on S4 having executed

D. S1 depends on S3 having executed

3.
Question 3

You can use futures in the Java Fork-Join framework (as discussed in Lecture 2.2) by implementing which class?

1 point

A. RecursiveAction

B. RecursiveTask

C. FutureTask

D. Future

4.
Question 4

How do you retrieve the value from a future in the Java Fork-Join framework (as discussed in Lecture 2.2) ?

1 point

A. Through the return value provided by explicitly calling a task’s compute() method.

B. By the return value of the fork() method of ForkJoinTask.

C. By the return value of the join() method of ForkJoinTask.

D. By the return value of the quietlyJoin() method of ForkJoinTask.

5.
Question 5

Consider the Pascal’s triangle implementation below, with a triangle containing R rows, with row *i* containing *i* entries. A triangle with R rows would therefore have 2R(R+1) total entries. Assuming a memoized parallelization strategy like the one below (and discussed in Lecture 2.3), how many futures would be created when computing Pascal’s triangle with R rows ?

```
final Map<Pair<Int, Int>, future<Integer>> cache = new ...;

future<Integer> choose(final int N, final int K) {

    final Pair<Int, Int> key = Pair.factory(N, K);

    if (!cache.contains(key)) {

        future<Integer> newFuture = null;

        if (N == 0) {

            // Root element case, zero gets

            newFuture = future {

                return 1;

            }

        } else if (K == 0) {

            // Left edge element case, one get

            newFuture = future { return choose(N - 1, K).get(); }

        } else if (N == K) {

            // Right edge element case, one get

            newFuture = future { return choose(N - 1, K - 1).get(); }

        } else {

            // Inner element case, two gets

            newFuture = future {

                return choose(N - 1, K - 1).get() +

                       choose(N - 1, K).get();

            }

        }
        cache.put(key, newFuture);
    }
    return cache.get(key);
}
// Scan across target row of Pascal’s triangle
final int targetRowLength = targetRowIndex + 1;
for (int i = 0; i < targetRowLength; i++) {
    result[i] = choose(targetRowIndex, i).get();
}
```

1 point

A. R

B. 2×R

C. R(R+1)

D. R(R+1)/2

6.
Question 6

Based on the same Pascal's triangle implementation above, if memoization and futures are used to compute a Pascal’s triangle of R rows, how many future get() calls must be made to compute the values for the R rows? Keep in mind the special cases of elements on the left and right edges of the triangle. You should ignore the get() calls on the second to last line (line 31) in the code example above; only consider the get()s necessary to compute the actual contents of the triangle.

1 point

A. R^2-R

B. 2R(R+1)

C. R(R+1)

D. 2×R

7.
Question 7

Supposed you had a List of Integers in Java: [3, 6, 8, 2, 1, 0]. Which of the stream-based programs below would be equivalent to the provided loop-based Java code? (Recall that Java streams were introduced in Lecture 2.4.)

```
List<Integer> output = new List<Integer>();
```

```
for (Integer i : input) {
    if (i >= 3) {
        output.add(i);
    }
}
```

1 point

A. input.filter(v -> v >= 3);

B. input.stream().filter(v -> v >= 3);

C. input.stream().filter(v -> v < 3);

D. input.stream().average();

8.

Question 8

Consider a search algorithm that uses many tasks to examine the search space in parallel. Every task that discovers a target value in the search space will set a global boolean flag to true (it is initialized to false). However, no task will ever read this global flag during execution, hence there will be no early exit of tasks. The flag will only be read after all tasks have terminated.

Such a program will exhibit which of the following? (Recall that data races, functional determinism, and structural determinism were introduced in Lecture 2.5.)

1 point

A. Functional and structural determinism, and no data race

B. Functional and structural determinism, with a data race

C. Functional but not structural determinism, with a data race

D. Structural but not functional determinism, with a data race

9.

Question 9

Now consider another search algorithm that uses many tasks to examine the search space in parallel. Each task increments a shared integer counter (e.g., count = count +1) when the search is successful.

Such a program will exhibit which of the following?

1 point

A. Functional and structural determinism, and no data race

B. Functional and structural determinism, with a data race

C. Functional but not structural determinism, with a data race

D. Structural but not functional determinism, with a data race

10.
Question 10

Finally, consider a variation of Question 8 in which every task that discovers a target value in the search space will set a shared global int variable to the location of the target value that was found (the variable is initialized to -1). However, no task will ever read the int variable during execution, hence there will be no early exit of tasks. The variable will only be read after all tasks have terminated. Note that, in general, there can be multiple possible locations for the target value, and we assume that any target location is acceptable in the final value of the int variable.

Such a program will exhibit which of the following?

1 point

A. Functional and structural determinism, with a data race

B. Functional and structural determinism, and no data race

C. Benign non-determinism, with a data race.

Honor Code Agreement