Concurrency & Parallelism

José Duarte

June 17, 2020

1

1.1 A

```
x = x + 1;
y = y + 1;
```
Listing 1: Thread 1

```
x = y + 1;
y = 2;
```
Listing 2: Thread 2

```
z = x + y;
```
Listing 3: Thread 3

```
x: 1, 2, 3, 4
y: 1, 2, 3
z: 0, 1, 2, 3, 4, 5, 6, 7
```

1.2 B

```
x = x + 1;
```
Listing 4: Thread 1

```
y = y + 1;
```
Listing 5: Thread 2

```
x = 5;
y = x + 1;
```
Listing 6: Thread 3

```
x: 1, 5, 6
y: 1, 2, 6, 7
(x, y): (1, 1), (1, 2),
        (5, 1), (5, 6),
        (6, 1), (6, 7)
```

2

False The `volatile` keyword does not stop multiple threads from modifying the variable. The expression `x++` can be broken down into a read-modify-write sequence, which can be interleaved between threads.

3

1. Safety
2. Liveness
3. Liveness
4. Safety
5. Safety

4

1. G
2. A
3. D
4. B
5. C
6. F
7. H
8. C

5

Because the barrier will ensure that all threads wait for each other.

6

6.1 a

The number of combinations is $3! = 6$ and the possible combinations are:

```
1 3 5
1 5 3
3 1 5
3 5 1
5 1 3
5 3 1
```

6.2 b

```
1 3 5 2 4 6
```

6.3 c

```
3 6 2 5 4 1
```

7

```
public class BarrierN {
    private int a, b;

    public BarrierN (int howmany) {
        b = howmany;
        a = 0;
    }

    public synchronized void arrive() {
        a++;
        if (a < b) {
            this.wait();
            return;
        }
        this.notifyAll();
    }
}
```

8

8.1 a

In the line 11, the program read the values $[20, 25, 30]$, before M_2 runs the `credit` statement, M_1 `transferTo` is able to execute completely and then M_2 `credit` statement runs, adding an old interest value.

8.2 b

The way it is written the program cannot deadlock, unless the client can have accounts that are references to other accounts.