For the example below, will reordering the five async statements change the meaning of the program (assuming that the semantics of the reader/writer methods depends only on their parameters)? If so, show two orderings that exhibit different behaviors. If not, explain why not.

1. `DataDrivenFuture left = new DataDrivenFuture();`
2. `DataDrivenFuture right = new DataDrivenFuture();`
3. `finish {
4.    async await(left) leftReader(left); // Task3
5.    async await(right) rightReader(right); // Task5
6.    async await(left, right)
7.         bothReader(left, right); // Task4
8.    async left.put(leftWriter()); // Task1
9.    async right.put(rightWriter()); // Task2
10. }`
Complete the phased clause below to implement the point-to-point synchronization shown above for m tasks (generalization of slide 11)

1. `finish () -> {
2.   final HjPhaser[] ph =
3.     new HjPhaser[m+2]; // array of phaser objects
4.   forseq(0, m+1, (i) -> { ph[i] = newPhaser(SIG_WAIT) });
5.   forseq(1, m, (i) -> {
6.     asyncPhased(
7.       ph[i-1].inMode(......),
8.       ph[i].inMode(......),
9.       ph[i+1].inMode(......), ()->{
10.       doPhase1(i);
11.       next();
12.       doPhase2(i); }) // asyncPhased
13.     }); // forseq
14. }); // finish