Worksheet: Computation Graphs for Async-Finish and Future Constructs

1) Can you write pseudocode with `async & finish` constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program. If not, why not?

2) Can you write pseudocode with `future & get` constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program. If not, why not?
1) Can you write pseudocode with `async-finish` constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program.

No. Finish cannot be used to ensure that D only waits for B and C, while E waits only for C.

2) Can you write pseudocode with `future async-get` constructs that generates a Computation Graph with the same ordering constraints as the graph on the right? If so, provide a sketch of the program.

Yes, see program sketch with dummy return values.
1. var A = future(() => {
    return "A";
});
2. var B = future(() => {
    A.get(); return "B";
});
3. var C = future(() => {
    A.get(); return "C";
});
4. var D = future(() => {
    B.get(); C.get(); return "D";
});
5. var E = future(() => {
    C.get(); return "E";
});
6. var F = future(() => {
    D.get(); E.get(); return "F";
});
7. F.get();

// Order of B.get() & C.get() doesn't matter