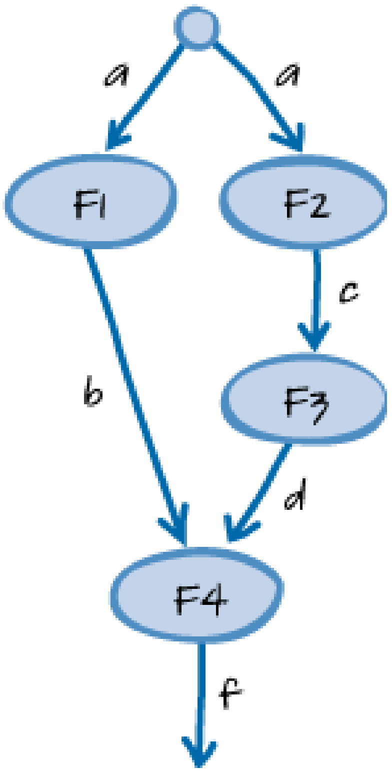
Quiz 9 C++ Tasks

1. What is a *task* in relation in concurrent and parallel programming?
   a) It is a thread group.
   b) It is a program in local memory and a collection of I/O ports.
   c) Tasks are sequential operations that work together to perform a larger operation.
   d) They are used to structure and design/decompose fine-grained parallel applications.

2. What are the advantages of *task dependency graphs*?
   a) Their use avoids race conditions and deadlocks.
   b) They help developers structure parallel systems into tasks.
   c) Once created, they can be implemented by C++ futures of by PPl (Parallel Patterns Library).
   d) They map easily to the object-oriented paradigm.

3. Consider the following task dependency graph:



   Which of the following statements are true (assume that there is no shared data)?
   a) Functions F1 and F2 can run in parallel.
   b) Function F4 must wait on function F3.
   c) Load balancing issues can occur if F1 is more computationally intensive than F2.
   d) F4 only needs to wait on F1 or F3.

4. Which statements accurately describe *Resource Acquisition Is Initialization* (*RAII*)?
   a) It avoids resource leaks without extensive use of `try/catch` blocks.
   b) It is used in the sense of the *finally* keyword in Java, for example.

c) It only works for heap-based objects.
d) It is possible to implement RAII using *active objects*.

5. Which of the following statements are relevant/true concerning RAII and C++ Concurrency?
   a) RAII ensures that mutexes are released when the relevant scope is exited.
   b) RAII ensures that race conditions do not occur.
   c) For an active object, it means that the object's embedded threads is joined in its destructor.
   d) `std::lock_guard` implements RAII.

6. What is an *active object* ?
   a) An object that has its own thread of control.
   b) It decoupled method invocation from method execution.
   c) It cannot be used in distributed/network environments.
   d) It synchronises concurrent method execution to ensure that only one method runs within the object.

7. What is `std::future` ?
   a) It provides a facility to store a value or an exception that is later acquired asynchronously.
   b) Provides a mechanism to access the result of an asynchronous operation with multiple threads being allowed to wait for the same shared state.
   c) A mechanism to access the result of an asynchronous operations.
   d) It is a mechanism for synchronous communication.

8. Which of the following statements pertaining to `std::async` are true?
   a) It starts an asynchronous function in a separate thread.
   b) It returns an instance of `std::future` containing the result of the computation.
   c) Exceptions are not stored in the future associate with `std::async`.
   d) *Lazy evaluation* is not possible.