1. Study Ben-Ari, Sections 7.1–7.5, 7.8, 7.9, 7.11.

2. Study the paper by Buhr, Fortier, and Coffin, *Monitor Classification*, Sections 1, 2, 3.

   http://www.cslab.pepperdine.edu/warford/cosc450/Monitor-Buhr.pdf

3. Study class handout: Notes on monitors.

   http://www.cslab.pepperdine.edu/warford/cosc450/cosc-450-Notes-on-Monitors.pdf

4. Study Sestoft, Chapter 16.

5. Implement the dining philosophers solution of Ben-Ari, Algorithm 7.5, in C++ with a monitor.
   The source file to modify is `Philosopher.cpp` in the `cosc450CppDistr` software distribution.

   Even though Algorithm 7.5 is not starvation-free, starvation will never occur because our runs are finite. Any two neighboring philosophers who conspire to starve the philosopher in the middle will eventually terminate, enabling the philosopher in the middle to continue.

   Your solution should consist of a monitor named `ForkMonitor` with the following methods.

   - `takeForks(int philID)`
     - The first statement after the `unique_lock` should output “Philosopher *p* is taking forks.” where *p* is the philosopher’s `philID`.
     - The last statement should output “Philosopher *p* is eating.”
   - `releaseForks(int philID)`
     - The first statement after the `unique_lock` should output “Philosopher *p* is releasing forks.” where *p* is the philosopher’s `philID`.
     - The last statement should output “Philosopher *p* is thinking.”

   The main program has the following specification.

   - `philosopherRun(int philID)`. Loop three times. Inside the loop, the philosopher should
     - random delay for 60ms,
     - pick up his forks,
     - random delay for 60ms,
     - put his forks down.
   - `main()`
     - Start five philosophers.
     - Join five philosophers.
     - Output “The Philosophers are finished eating.”

   Run your program several times to insure that it produces different results each time and does not deadlock. Show the following in the documentation section at the bottom of the source file.

   (1) Copy the output from one of the runs and paste it into the documentation section.
   (2) From the output in (1), what is the relationship between the statements “Philosopher *p* is releasing forks” and “Philosopher *p* is thinking”?
   (3) C++ uses Mesa semantics. Would the same relationship hold if the monitor used Hoare semantics?

(4) Explain your answer to (3).
(5) Does the same relationship in (2) hold between the statements “Philosopher *p* is taking forks” and “Philosopher *p* is eating”?
(6) Explain your answer to (5).
(7) From the output in (1), list the first eight philosophers, in chronological order, who started to eat.
(8) From the output in (1), identify the first philosopher who was blocked trying to pick up his fork.

For this problem, hand in

`Philosopher.cpp`