1. Study Ben-Ari, Sections 6.1–6.10.

2. Prove (119) Monotonicity of $\Diamond$: $\quad \Box(p \Rightarrow q) \Rightarrow (\Diamond p \Rightarrow \Diamond q)$.
   Hint: Start with the entire expression and get it equivalent to *true*. Use (3.59) three times. Then, eliminate $\Box$ with (60) Dual of $\Box$. Manipulate the resulting expression containing $\Diamond$ as the only temporal operator using (3.47b) De Morgan, (52) Distributivity of $\Diamond$ over $\vee$, (3.44b) Absorption, (3.28) Excluded middle, and (3.29) Zero of $\vee$.

   You may hand in the proof on paper or electronically. If you hand it in electronically it must be a PDF document named

   `a22written.pdf`

   per the instructions for your course.

3. Implement the bounded buffer producer-consumer problem with C++ counting semaphores.
   See Ben-Ari, Problem 6.9. The source file to modify is `Algorithm-6-19.cpp` in the `cosc450CppDistr` software distribution.

   Implement the bounded buffer as a circular queue having five elements.

   The `producer` thread should:

   - Execute in a loop 15 times producing the values 10, 20, ..., 150 to store in the circular queue.
   - Print the message "Produced *x*" on a new line where *x* is 10, or 20, etc. just after leaving its critical section (after the `signal()` statement).

   The `consumer` thread should:

   - Execute in a loop 15 times consuming the values.
   - Print the message "Consumed *x*" on a new line where *x* is the value retrieved from the buffer just after leaving its critical section (after the `signal()` statement).

   To test the efficacy of your code, put random delay statements inside and outside the critical sections. To insure that there is no interleaving in your output statements, which are not atomic, incorporate a `mutex` binary semaphore to implement mutual exclusion for them.

   In addition to handing in your correct program, do an experiment by commenting out the `wait` and `signal` statements for accessing the buffer. Do *not* comment out the `mutex` operations on the output statements. Then run the program a few times and select a run that exhibits erroneous behavior. Copy the output of the run into the comment field provided at the bottom of the `.cpp` source file. Below the printout of the erroneous run, explain (1) where the first error occurred, (2) what the error was, and (3) what happened to cause the error.

   For this problem, hand in

   `Algorithm-6-19.cpp`

   CAUTION: Do not hand in your erroneous listing with the `wait` and `signal` methods commented out. Hand in your correct solution of the bounded buffer producer-consumer problem.