Race Condition:

- What is race condition?
    - A race condition is a situation in which the output of a process relies on the result of another event which may or may not finish executing in time.
- Why is a race condition difficult to reproduce and debug?
    - The end result of the program depends on the runtime of events that are variable and extremely difficult to plan around.
- How can it be fixed?
    - It is usually better to avoid these bugs rather than attempting to fix them, but they can be solved by making sure that processes do not share memory, like in Assignment three, when we used different code to make sure that the threads did not all use the same memory and add cumulatively.
- Summarize the Parallel Programming Patterns section in your own words
    - Parallel programming makes use of repeated effective patterns, which makes it relatively easy to learn. There are strategies, and concurrent execution mechanisms. When considering strategies, there are the subcategories of algorithms and implementation: algorithms being the actual methods used to solve the problems and implementation being the manner in which you use the algorithms, for example what processes can be done at the same time. Concurrent execution mechanisms are code patterns that can be used by programmers. They include process/thread control and coordination. Process/thread control patterns cover how specific processing units execute at run time. Coordination patterns outline how the above units interact with each other to produce correct output. It can be further broken down into message passing and mutual exclusion, which can be realized by using OpenMp or MPI. There is also a third type of implementation that uses both patterns.

- Compare collective communication (reduction) and collective synchronization (barrier)
    - Both collective communication and collective synchronization involve a group of processes that come together to achieve an end result.

- Compare master-worker and fork-join
    - Both the master-worker and fork-join program structures has a main computation that creates small tasks that are worked on by worker threads in the thread pool and when they are done, they come back to the main computation that started those small tasks.

- Where can we find parallelism in programming?
    -
- What is dependency and what are its types (provide one example for each)?
    - Dependency is when one the execution of one process depends on the output of another process.
    - In true or flow dependence, the second statement depends on the first.

        - x = 5;
          y = 3 * x;
        - In this example, the output of the second statement is dependent on whether or not the first has already been executed.
    - In output dependence, the output of each individual statement is unchanged by the order of execution, but the final outcome of the execution of both statements does depend on order.
        - x = 5;
          x = 6;
        - The order in which these statements are executed changes the final value of x.
    - In anti-dependence, the first statement depends on the second.
        - x = 3 * y;
          y = x;
        - The first statement would have a different output if the second statement were executed first.
- When a statement is dependent and when is it independent (provide two examples)?
    - Two statements are independent when their outputs are unchanged by the order in which they are executed. For example:
        - x = 2;
          y = 2 * 2;
    - Two statements are dependent when the order in which they are executed changes their outputs. For example:
        - x = 2;
          y = x * x;
- When can two statements be executed in parallel?
    - Two statements can be executed in parallel only if they are independent of each other.
- How can dependency be removed?
    - In some cases, one can remove dependency by rewriting code to order statements differently or by only using the base variable to set other variables instead of using intermediary variables (ex. A=5; B=5; instead of A=5; B=A), or by removing statements altogether.
- How do we compute dependency for the provided loops and what types of dependency are they?
    - To compute dependency you have to compare the IN and OUT sets. B