New Parallel Algorithms Conclusion Solutions

Disadvantages of Execution Policies

- Give some disadvantages of execution policies
  - Some compilers do not support execution policies
  - Some compilers have not fully implemented execution policies
  - If the execution policy makes the algorithm start up new threads, this will add overhead

When to Use an Execution Policy?

- Suggest some situations where it would ***not*** be desirable to use an execution policy
  - The code has to be portable to other compilers
  - The task is essentially sequential
  - Operation order is important
  - The algorithm call throws exceptions and immediately terminating the program is not acceptable
  - If the algorithm with execution policy starts new threads and preventing data races adds more overhead than not using the execution policy

Which Execution Policy to use?

- Describe when you would use each of the execution policies, and explain why
- Sequenced execution is mainly used for debugging
  - It allows out of order execution, and terminates the program on exceptions, but it is guaranteed to only use one thread
- Parallel unsequenced execution, whenever possible
  - It has the most potential to improve performance
  - However, it requires that data races cannot occur, and the code does not modify shared state (memory allocation/deallocation and thread synchronization)

Which Execution Policy to use?

- Parallel execution, when vectorization is not safe, because the code modifies shared state
  - Data races cannot occur
- Unsequenced execution
  - Requires a compiler which supports C++20
  - Can be used in single threaded programs, or if parallel execution is not safe
  - Code must not modify shared state