New Parallel Algorithms Solutions

std::accumulate

- Briefly describe the std::accumulate function and write a program to demonstrate its use
    - std::accumulate returns the sum of the values of its elements
    - It takes the initial value of the sum as an argument
    - It also takes an optional callable object, which will be used instead of the + operator in the calculation
- Write another version of this program which uses a parallel execution policy

std::partial_sum

- Briefly describe the std::partial_sum function and write a program to demonstrate its use
  - std::partial_sum uses the elements of a container to populate another container
  - The nth element in the target vector will be the sum of the first n elements in the source vector
  - e.g. {1, 2, 3, 4} gives {1, 1+2, 1+2+3, 1+2+3+4}
- Write another version of this program which uses a parallel execution policy
- Using a different function, write another version of this program which uses a parallel execution policy

std::adjacent_difference

- Another algorithm in the <numeric> header is std::adjacent_difference
- This populates a vector with the difference between successive elements
  - e.g. {1, 2, 3, 4} gives {2-1, 3-2, 4-3}
- However, it was not necessary to create a new function to add support for execution policies to adjacent_difference. Why was this?
  - The specification of std::adjacent_difference does not require sequential ordering, unlike std::accumulate
  - The existing implementation can be directly made parallel and does not need to be reimplemented