Packaged Task Solutions

std::packaged_task

- Briefly describe the std::packaged_task class
  - std::packaged_task is a wrapper class which contains a callable object and a promise
  - The callable object is passed to the packaged_task constructor. Its signature must match the template parameter of the packaged_task instance
  - A std::packaged_task instance is itself a callable object
  - Normally, it is passed to an std::thread constructor, along with any arguments to its callable object member
  - When the thread runs, the packaged_task's callable object is invoked. The return value from this call is stored in the packaged_task's promise
  - We can get this result by calling get_future()

packaged_task example

- Write a program which creates a packaged_task. The packaged_task's callable object member will take two int arguments and add them together. The program will print out the result
- Write another program which performs the same addition, but uses an explicit promise and future instead of a packaged_task (similar to the "Producer-Consumer" example in the Promises lecture)

Thread container

- Imagine you want to create a container whose elements are runnable threads. Which class would you use for the elements?
    - std::packaged_task would be a good choice because the thread objects can be made to start running at a time of our choice
    - std::thread could also be used, but the thread starts running as soon as the object is created. In some applications this is a disadvantage