Thread Pool with Work Stealing Solutions

Work Stealing

- Briefly describe what is meant by "work stealing"
  - Work stealing is a refinement of "work sharing"
  - Each thread in the thread pool has its own queue of tasks
  - However, when a thread has no tasks, it does not wait to be given one
  - Instead, it takes a task from another thread's queue
- Give an example of a situation in which this might be useful
  - One of the threads is performing a task which takes a long time to complete
  - The tasks on this thread's queue would have to wait until this task completes
  - With work stealing, another thread which has no work takes this task and executes it
  - The "thief" thread executes this task before the long-running task completes

Work Stealing Strategy

- Briefly outline a strategy which could be used to implement task stealing
  - If a thread's queue is empty
    - Do not wait for a task to arrive on the queue
    - Choose another thread's queue at random
    - If there is a task on that queue, pop it and execute it
    - Otherwise, choose a different thread's queue at random
    - Continue until it finds a task to perform
  - If all the queues are empty
    - Pause for a while
    - Then repeat the process

Thread Pool with Work Stealing

- (Optional) Implement your answer to the last question
  - The solution will be given in the next lecture!