Parallel Computing: Performance Overview Quiz by CCL

1. Which approach should we use to benchmark runtime across languages in Julia? (2:49)
   - ○ The `@time` macro
   - ○ BenchmarkTools
   - ○ `time_ns()`
   - ○ A stopwatch

2. What is NOT a possible cause of inconsistencies in runtime taken, when summing elements in a `rand(1:10000)` dim array? (3:00)
   - ○ Using an unreliable library for timing
   - ○ The random seed for array dimensions
   - ○ Competition with other computer process
   - ○ None of the above

3. What of the following is NOT a feature of BenchmarkTools' @benchmark macro? (4:26)
   - ○ Memory estimate
   - ○ Allocs estimate
   - ○ Remaining RAM
   - ○ Mean time

4. Which of the following is NOT true about benchmarking Python and C in Julia? (time stamp: 7:41)
   - ○ C code in IJulia is commonly executed with Libdl
   - ○ Python code in IJulia is commonly executed with Pycall
   - ○ Python built-in functions are slower than Numpy
   - ○ Numpy is implemented in and callable from Python

5. Which of the following is a reason why Julia built-in and Python Numpy functions are so much faster than Python? (15:53)
   - ○ Python is a dynamically-typed language and checks variable types during runtime
   - ○ Numpy and Julia exploit parallelism
   - ○ Python interprets code at runtime to know which functions to dispatch
   - ○ All of the above

6. Which of the following is NOT true about allowing floating point associativity in Julia? (16:19)
   - ○ Uses packages Distributed and DistributedArrays
   - ○ Puts to work multiple cores or computers for a single task
   - ○ Is an example of parallel computing
   - ○ Is only possible on GPU machines

7. What is the main difference between parallelism with SIMD vs DistributedArrays? (26:12)
   - ○ Only the latter allows for floating point associatvity
   - ○ The former is performed on CPUs, and the latter on GPUs
   - ○ Parallelism with SIMD is generally faster than with DistributedArrays
   - ○ SIMD performs single-processor parallelism, while DistributedArrays perfroms multi-process parallelism

Answers: B, D, C, D, D, D, A