1. 1. Pascal supports unified memory for easy transfer of data in CUDA programs
   2. A new high bandwidth bus called NVLink is supported for communication between the GPU and CPU
   3. Half precision floating point operations can run at 2x the speed of single precision
   4. The number of registers per CUDA core has doubled
   5. The amount of shared memory has increased
   6. A new dynamic load balancing system has been implemented to keep the GPU as close to maxed-out as possible
   7. CUDA Compute Capability 6.0 is now supported
   8. “High Bandwidth Memory 2” is now used to increase memory throughput.
2. While a thread can use that many registers, there are only so many total registers per stream multiprocessor, which means that the more registers used per thread in a block, the fewer blocks can be run at a time in the SM, thus reducing the total number of threads running at once.
3. Threads within a block can share data using “shared” variables declared within the kernel with the prefix “\_\_shared\_\_”
4. No. All threads in a warp are either running the same instruction or sleeping. They all start and finish at the same time regardless of the number of instruction cycles any particular one needs.
5. An if-then-else block is executed in the following manner:
   1. All threads calculate the value of the conditional
   2. The “then” block is executed
      1. If the conditional result was true for a thread, the thread executes these instructions
      2. If the conditional result was false, the thread sleeps
   3. The “else” block is executed
      1. If the conditional result was true for a thread, the thread sleeps
      2. If the conditional result was false, the thread executes these instructions
   4. All threads continue executing the remaining instructions
6. Unified virtual addressing keeps variables stored on the host (computer/RAM) consistent with variables of the same name and address in the device (GPU). This is done automatically as needed without the programmer needing to explicitly copy data back and forth. It allows the same variables/pointers/data to be used in both the host and device code without having to worry about data consistency or valid host/device addresses.
7. Running the calculation on the CPU with OpenMP, I get a time of about 0.04 seconds. On the GPU, the same calculation is complete in 0.0006 seconds.
8. Potential energy: -1.02685e+10
9. Potential energy: -1.37127e+12