Part 1: CUDA Warm-Up 1: SAXPY

To gain a bit of practice writing CUDA programs your warm-up task is to implement the SAXPY function (`result = scale*X+Y`, where `X`, `Y`, and `result` are vectors of N elements and `scale` is a scalar value).

Please finish off the implementation of SAXPY in the function `saxpyCuda` in `saxpy.cu`. You will need to allocate device global memory arrays and copy the contents of the host input arrays `X`, `Y`, and `result` into CUDA device memory prior to performing the computation. After the CUDA computation is complete, the `result` must be copied back into host memory.

As part of your implementation, add timers around the CUDA kernel invocation in `saxpyCuda`. After your additions, your program should time two executions

- The provided starter code contains timers that measure **the entire process** of copying data to the GPU, running the kernel, and copying data back to the CPU.
- Your timers should measure only the time taken to run the kernel. (They should not include the time of CPU to GPU data transfer or transfer of results back to the CPU.)

When adding your timing code, be careful: The CUDA kernel's execution on the GPU is asynchronous with the main application thread running on the CPU. Therefore, you will want to place a call to `cudaDeviceSynchronize` following the kernel call to wait for completion of all CUDA work on the GPU. This call to `cudaDeviceSynchronize` returns when all prior CUDA work on the GPU has completed. (Without waiting for the GPU to complete, your CPU timers will report that essentially no time elapsed!) Note that in your measurements that include the time to transfer data back to the CPU, a call to `cudaDeviceSynchronize` **is not** necessary before the final timer (after your call to `cudaMemcpy` that returns data to the CPU) because `cudaMemcpy` will not return to the calling thread until after the copy is complete.

Question 1. What performance do you observe compared to the sequential CPU-based implementation of SAXPY? Compare and explain the difference between the results provided by two sets of timers (the timer you added and the timer that was already in the provided starter code). Are the bandwidth values observed roughly consistent with the reported bandwidths available to the different components of the machine? (Hint: You should use the web to track down the memory bandwidth of the NVIDIA GPU you've used for your experiment, and the maximum transfer speed of the computer's PCIe bus.