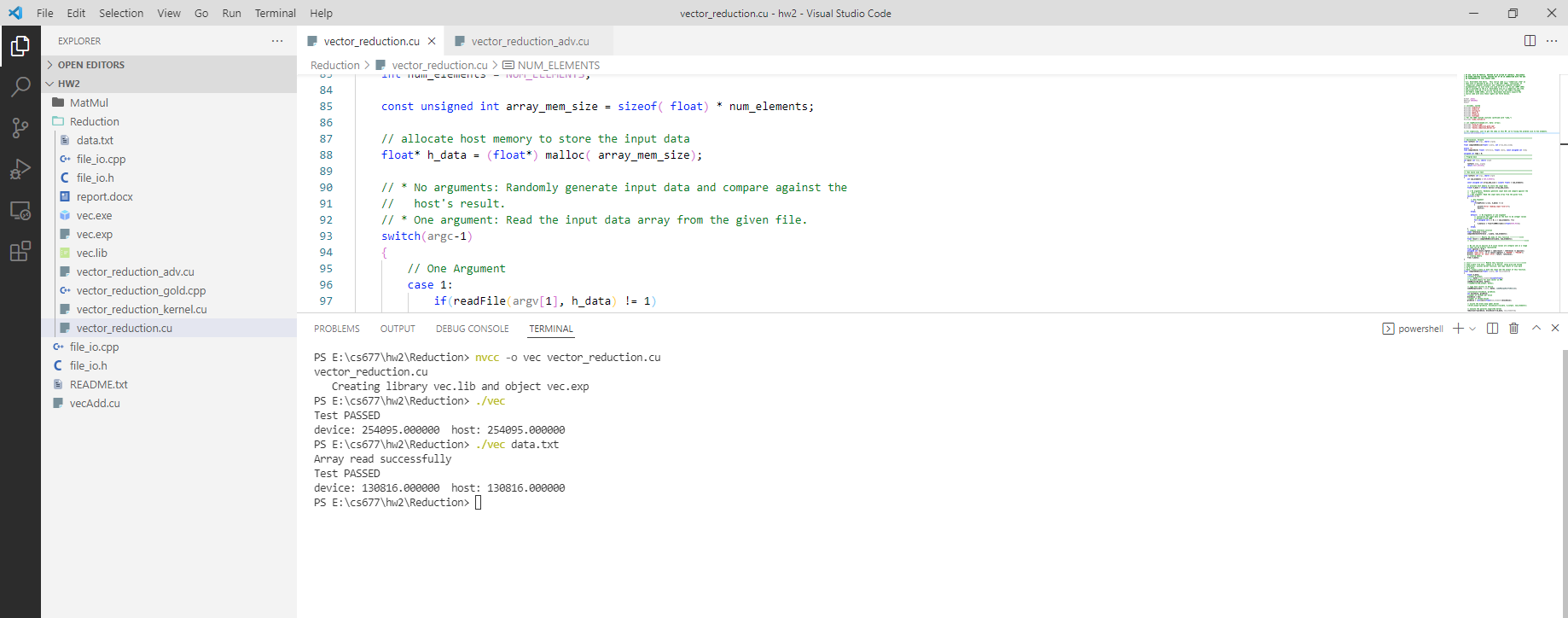
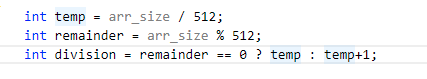
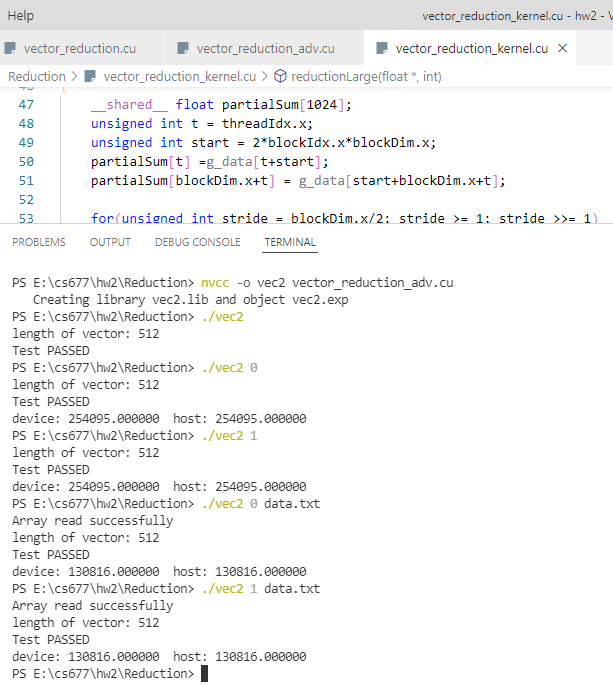
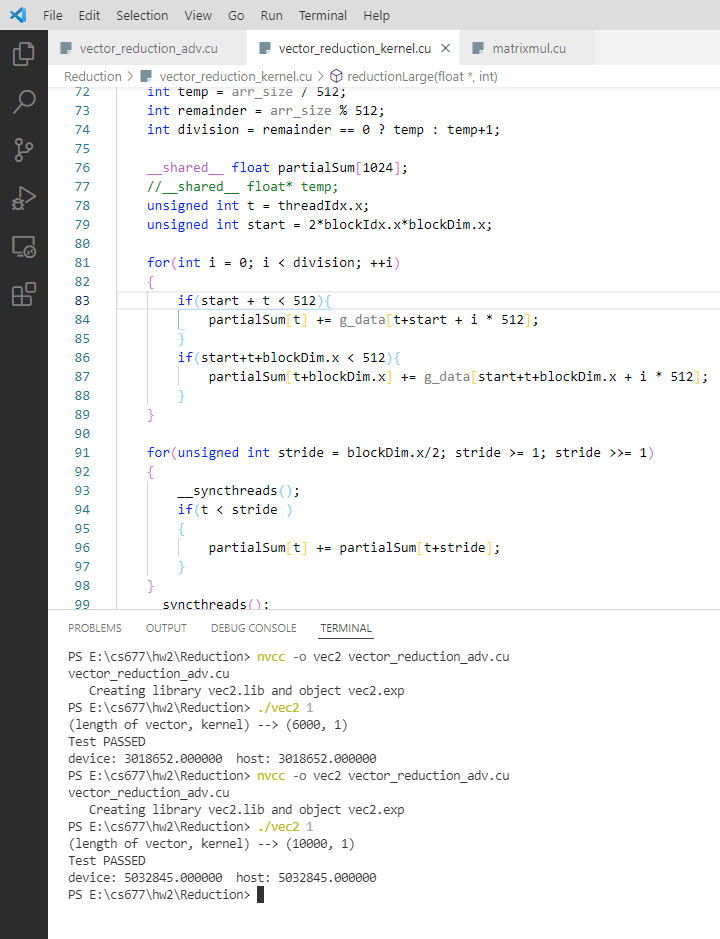
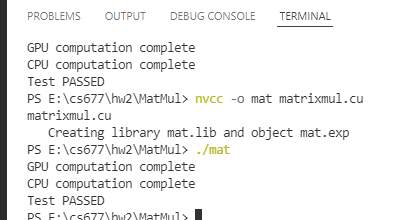
1. Problem1
   1. 2 times synchronize
   2. Maximum number of “real” operations: 8; Minimum number of “real” operations: 0; Average number of “real” operations: 4.
2. Problem2
   1. I add another kernel in ‘vector\_reduction\_kernel.cu’ named reductionLarge() which take case of vector size greater than 512.
   2. In reductionLarge() function, it can find out how many times this vector’s size is 512. Then, I use a for-loop to load data to threads and calculate the sub-value. After the calculation, storing them into a shared memory.
   3. 
   4. 
3. Problem3
   1. 
4. Problem 4
   1. 1024 \* 256 = 262,144
   2. Two global memory loads and one global memory store for each thread.
   3. 256 \* 2 + (128 + 64 + 32 + 16 + 8 + 4 +2+1) \* 3 = 1277
   4. None
   5. 128 / 2 / 2 / 2 / 2 / 2 / 2 / 2 / 2 = 0; So, there is 8 iterations of the for loop in total and 5 iterations will have branch divergence.
   6. There are 256 times to write back to the global memory. By reduction, we can store all value to thread 0 or the last thread then let that thread access the global memory, which can eliminate 1024 \* 255 accesses.