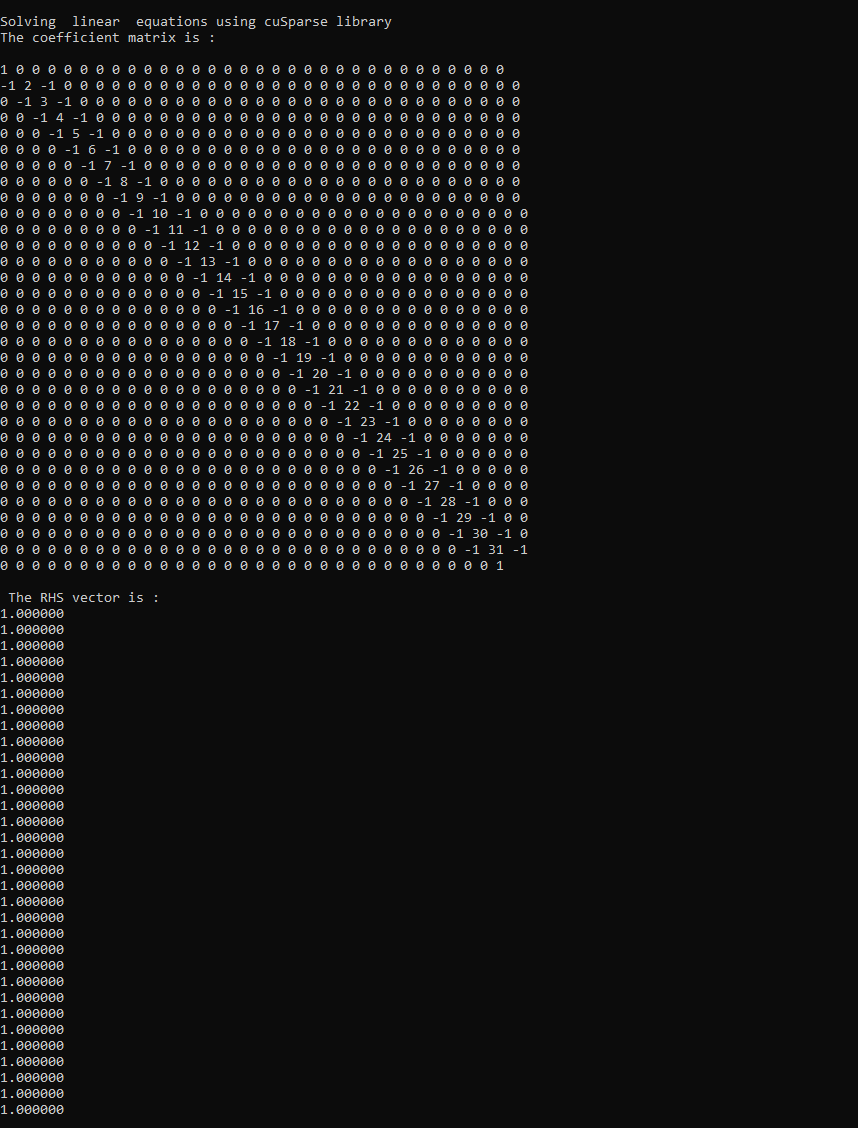
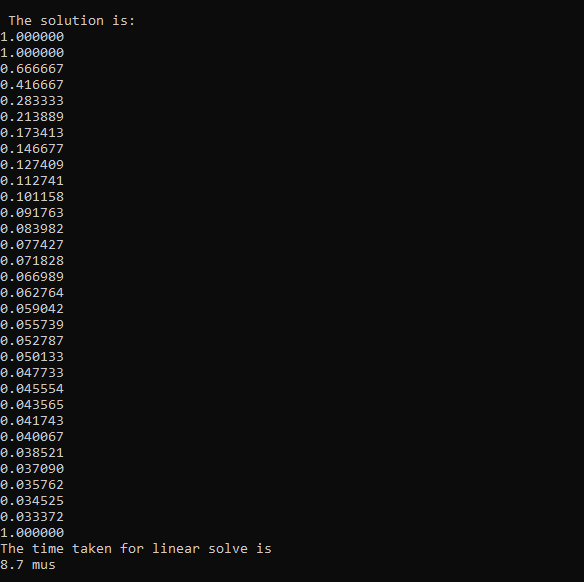
**Module 8 Question 1**

**Nandan Joshi**

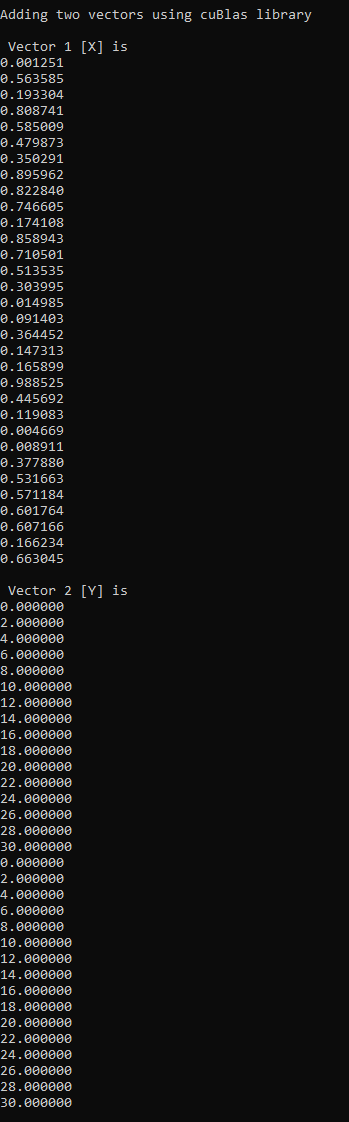
**1] Input Size 32**

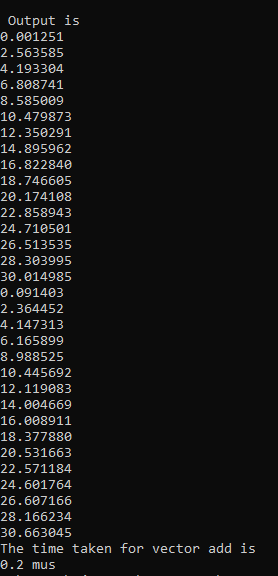
**CuSparse Library equation solve :**





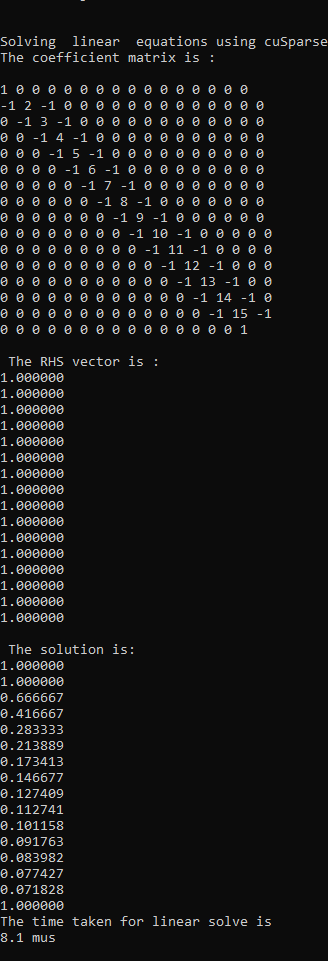
**Vector Addition using cuBlas**



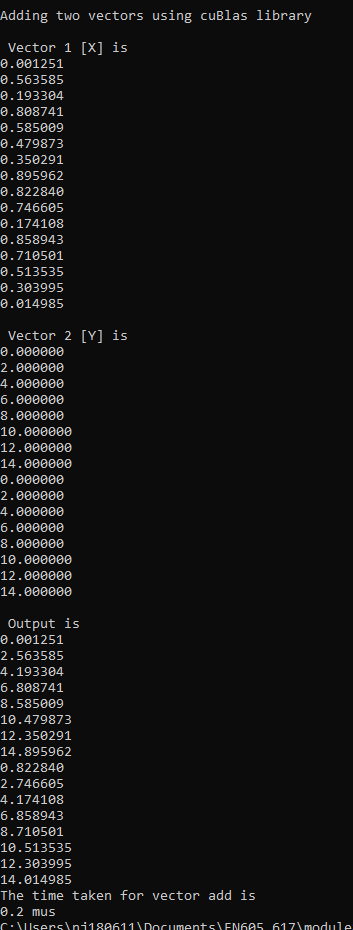


**1] Input Size 16**

**CuSparse Library equation solve :**



**Vector Add**

Is 

**Performance Metrics :**

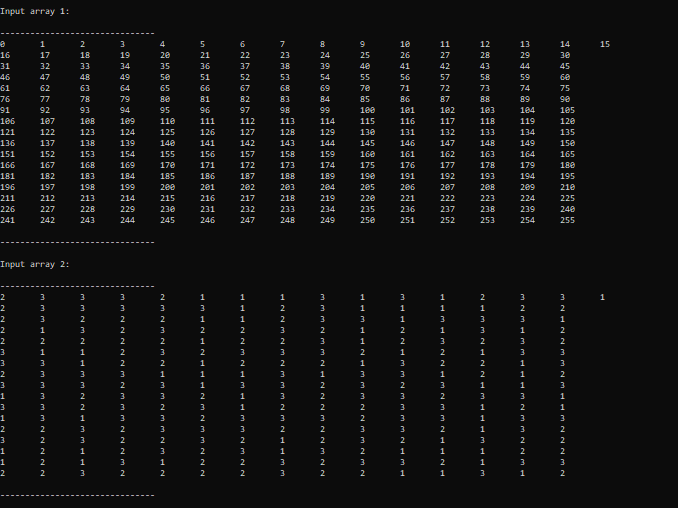
The runtime for the linear equation solve is plotted below :

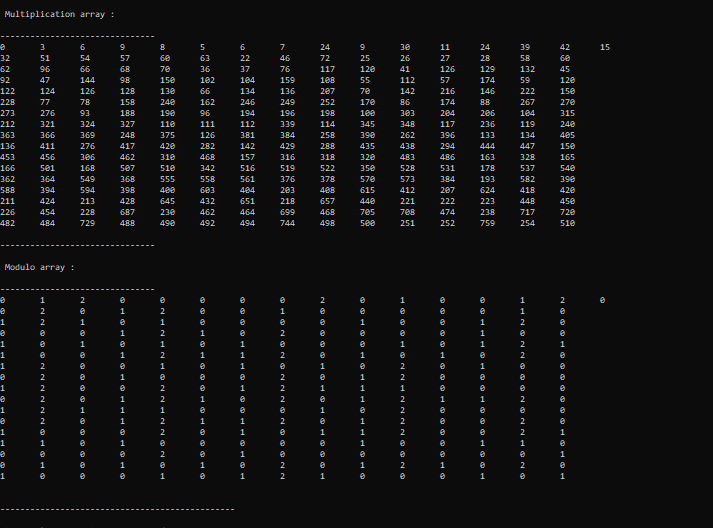
It offers some very instructive insights into the behaviors of the linear solve using GPUs. The runtime is fairly constant till 4096. After that, it follows an O(N) behavior. 4096 input size roughly corresponds to 128kB of memory (plus any additional value required by the buffer. So, it seems that the GPU is able to perform perform linear solve with full parallelism only till a limited input size After that, the parallelism breaks down (most likely due to insufficient GPU resources), and it starts following an O(N) behaviour. As the given matrix system is tridiagonal, O(N) is not such a great performance because even a serial tridiagonal solve executes in O(N). However, the actual execution time by the GPU may still be better, thus providing some improvement over the serial code on CPU.

The vector solve, on the other hand, demonstrates full parallelism as seen below –

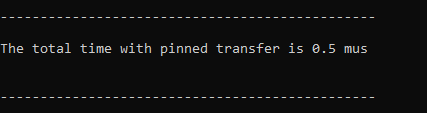
There is absolutely no change in execution time with larger input sizes. As a vector add instruction, in general, is fully parallelizable, we can see the improvement that an SIMD hardware with enormous number of threads brings about.

**Pinned transfer multiplication and modulo :**



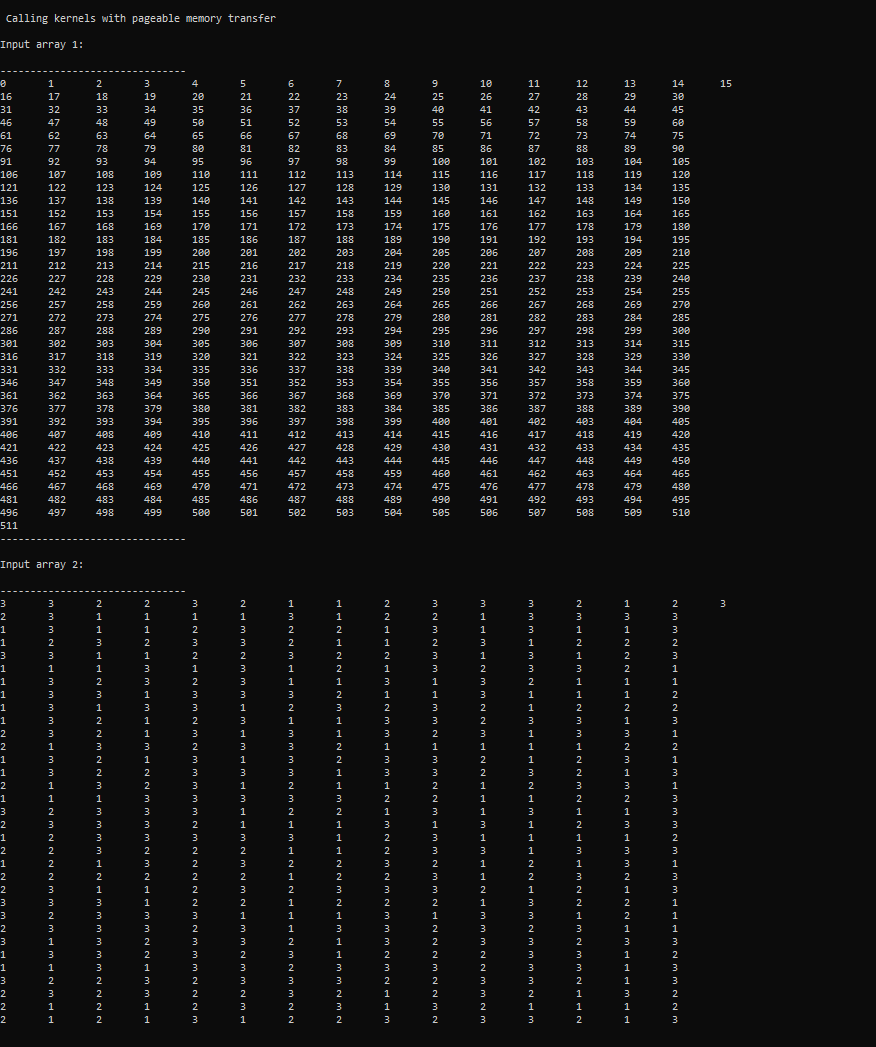
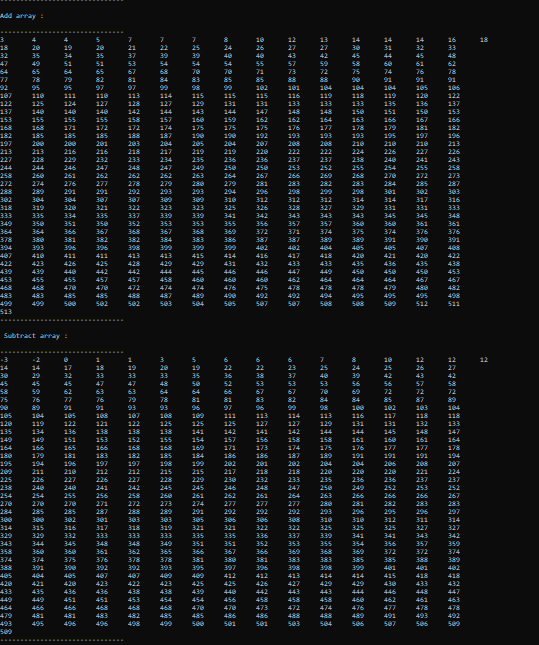


**Time Taken :**

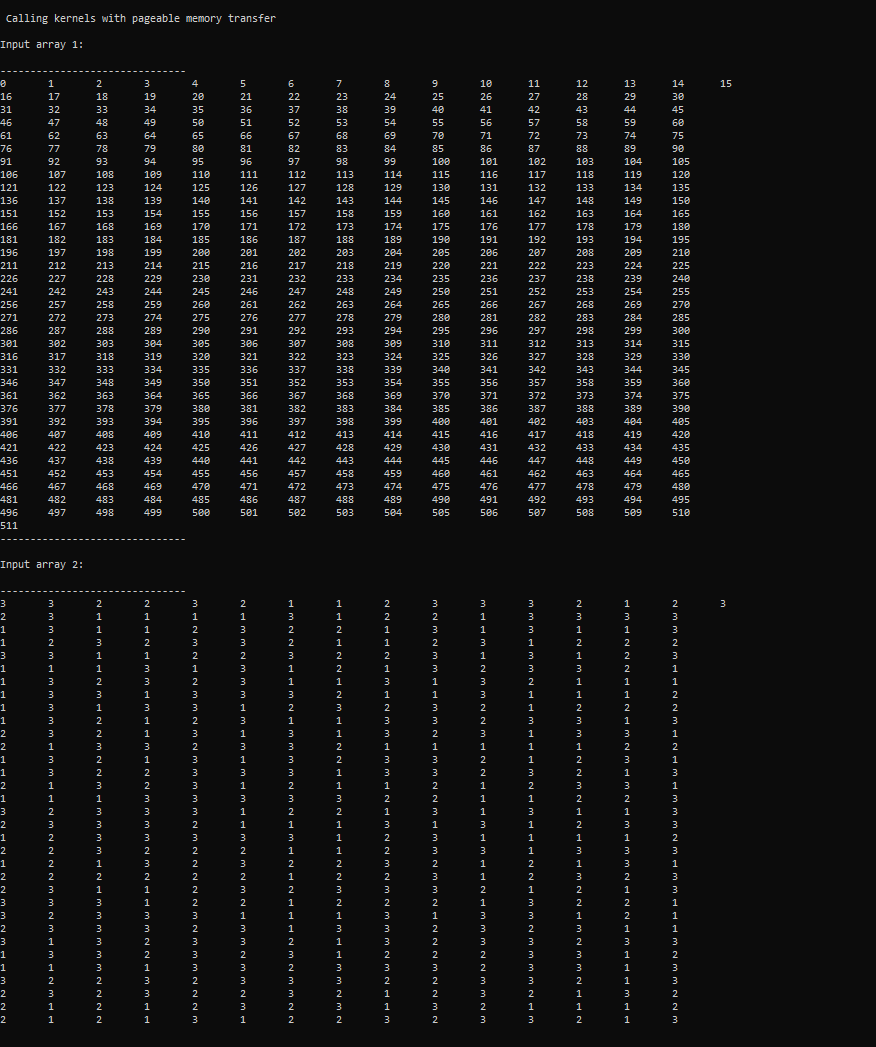


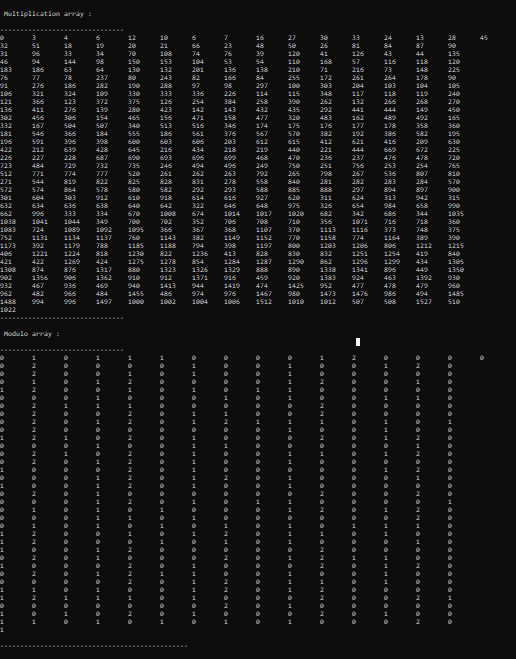
**1] No of threads: 512 thread size per block : 128 (4 blocks)**

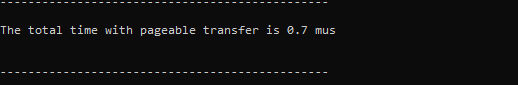
**Pageable transfer addition and subtraction :**

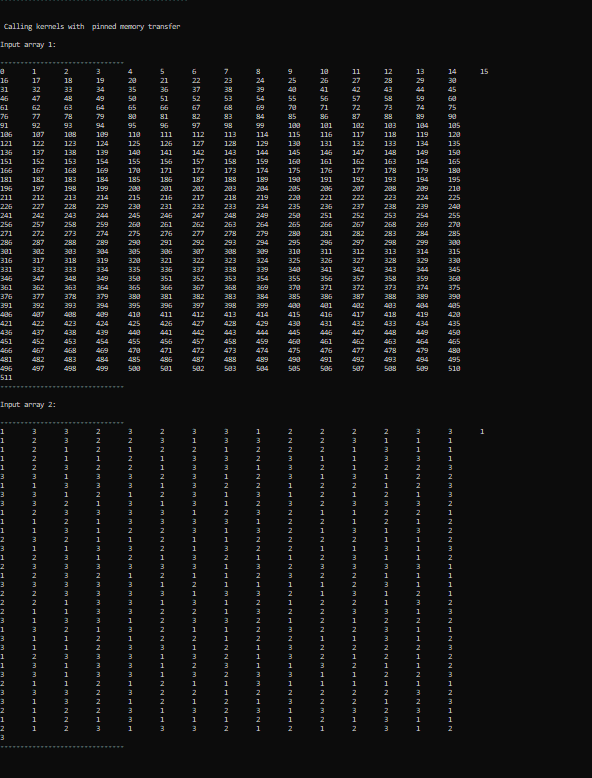
Pageable transfer multiplication and modulo

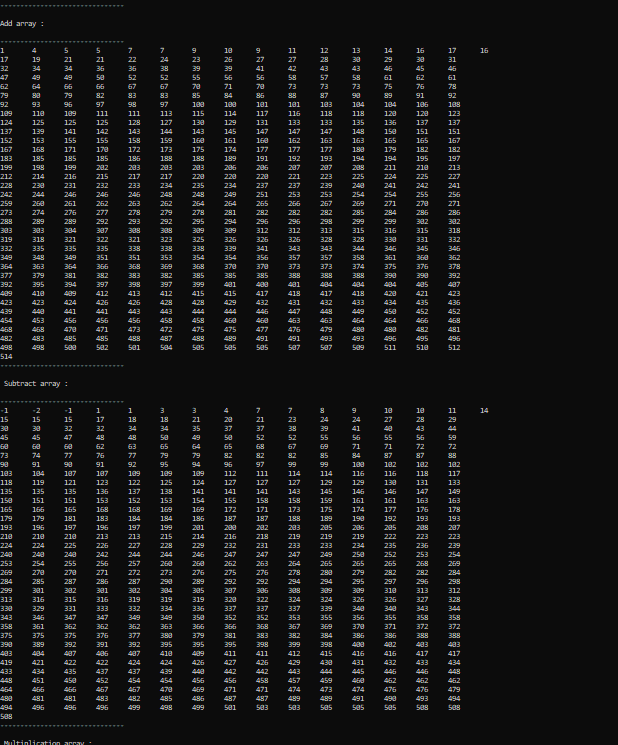




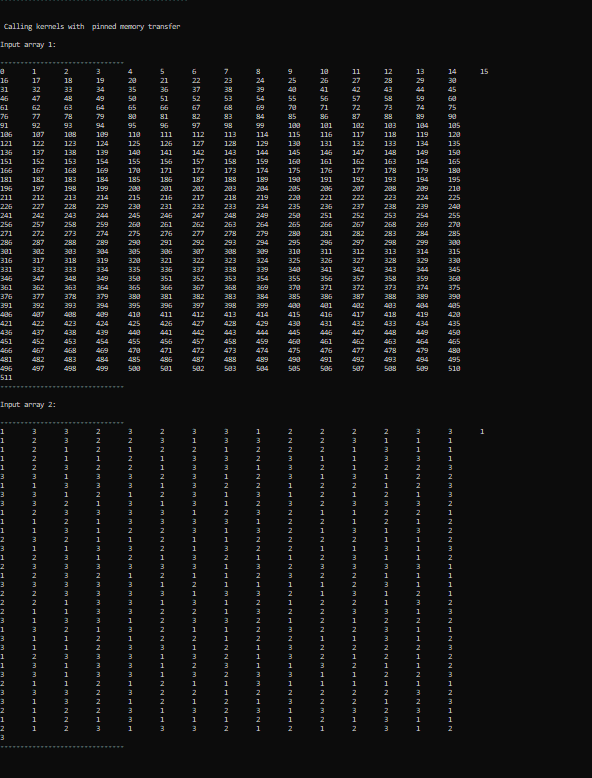


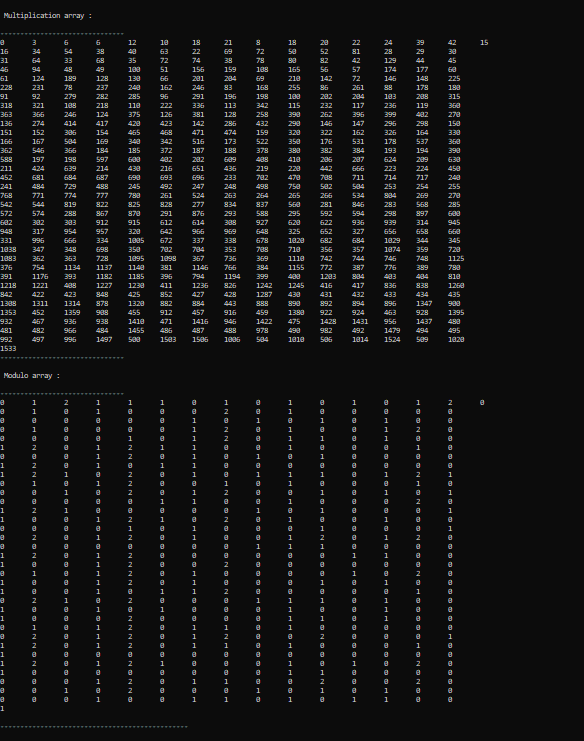
**Pinned transfer addition and subtraction :**

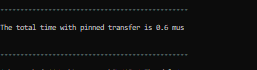




Pinned transfer multiplication and modulo :

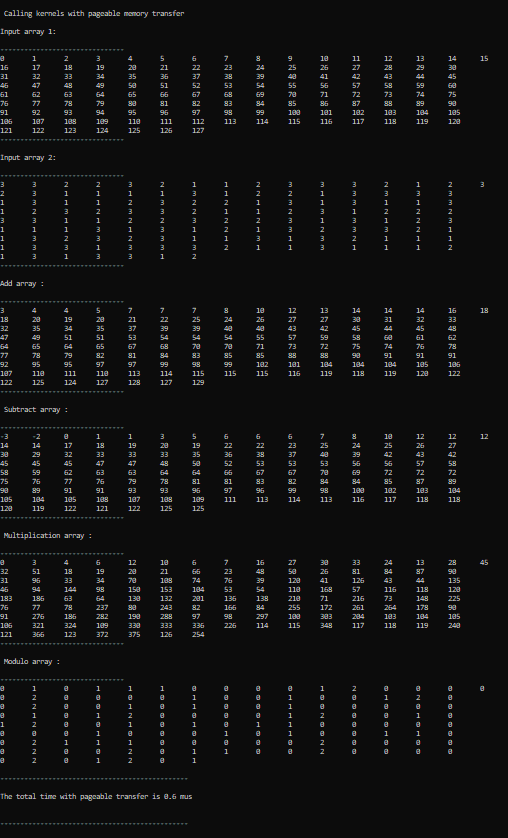




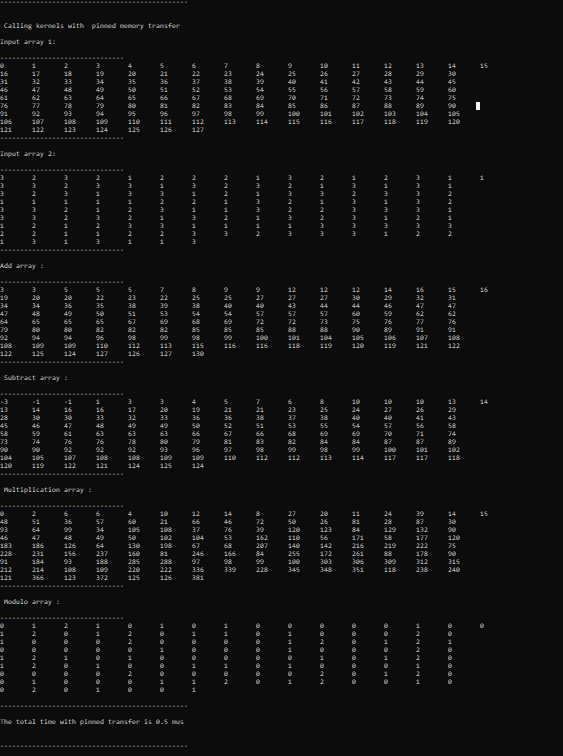


**1] No of threads: 128 thread size per block: 64 (2 blocks per thread)**

**Pageable :**

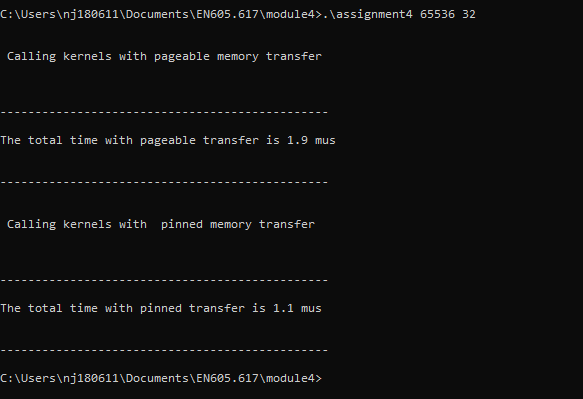


Pinned:



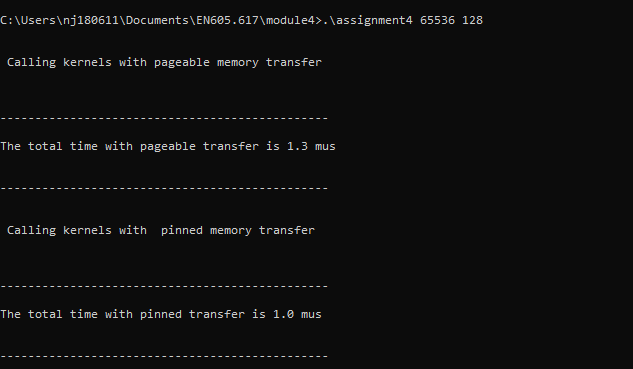
**4] No of threads 2^16 (65536), threads per block : 32**

**Note – Output of the kernels has been hidden due to the large size**



**5] No of threads 2^16 (65536), threads per block : 128**

**Note – Output of the kernels has been hidden due to the large size**



**6] No of threads 2^24 (16777216), threads per block : 128**

**Note – Output of the kernels has been hidden due to the large size**

