Assignment 4: Matrix Transpose

Recall the shared memory implementations of matrix transpose operation. It focused primarily on optimizations involving shared memory with square tiles. Consider a more general scenario where tiles are rectangular in nature.

Implement a transpose kernel which performs the following memory operations:

A. Write to a shared memory row with each warp to avoid bank conflicts.
B. Read from a shared memory column with each warp to perform a matrix transpose.
C. Write to a global memory row from each warp with coalesced access.

Implement a CUDA program which takes as input:
i) the number of test cases, for each test case,
ii) value of N and
iii) N lines of floating-point values where each line contains N space-separated floating-point numbers.

This represents the NxN input matrix. Assume that N is divisible by 32. Your program should print the transposed matrix as N lines of N space-separated floating-point values for each test case.

Expect N to be of the order 512,1024,2048, etc. Your task would be to generate random floating-point matrices of this order and test your implementation while adhering to the input format discussed above. The code should be general enough to handle large matrices. Your implementation should be generic enough to identify suitable grid block dimensions and shared memory dimensions. The shared memory should be rectangular in nature and of the dimensions W x (2*W).

Depending on what GPU you are working on, identify first the maximum shared memory size per SM. Accordingly, decide for your given input matrix what the value of W should be. Ensure that you print your output matrices using %.2f format.