Question 1:
Write Cuda code for computing matrix-matrix product of the form C=A*B where A is an m*k (double) matrix and B is k*n (double) matrix and C is an m*n (double) matrix. The code should be tiled and should use shared-memory. Make sure that the code works for matrices of all sizes not just square matrices. Follow the given template code.

Answer:
I wrote the kernel for this question in kernel_nt.cu file attached here. Table 1 is a summary of time and GFLOP/S for different values of M, N, and K. To compile the program, use: "`nvcc -O3 matmul_double.cpp kernel_nt.cu -o matmul_double`"

M	N	K	Time (ms)	GFLOPS
16	16	16	0.022400	0.354286
16	16	256	0.104736	1.249007
16	16	512	0.198976	1.316179
256	256	16	0.024224	83.867899
256	256	256	0.294656	113.654210
256	256	512	1.137920	58.917434
512	512	16	0.021760	373.458825
512	512	256	0.585600	228.749282
512	512	512	2.121504	126.407164
4095	4097	125	1.065376	110.234092

Question 2:
Write Cuda code for computing matrix-matrix product of the form C=A*BT where A is an m*k (double) matrix and B is n*k (double) matrix and C is an m*n (double) matrix; without explicitly transposing B. The code should be tiled and should use shared-memory. Make sure that the code works for matrices of all sizes, not just square matrices. Modify the given template code and use it.

Answer:
I wrote the kernel (`matmul_double_t<<<..>>>`) for this question in kernel_nt.cu file attached here. Table 1 is a summary of time and GFLOP/S for different values of M, N, and K.
My code works. However, I have a challenge with error checking.
To compile the program, use: "`nvcc -O3 matmul_double_t.cpp kernel_nt.cu -o matmul_double_t`"

M	N	K	Time (ms)	GFLOPS
16	16	16	0.022784	0.348315
16	16	256	0.103168	1.267990
16	16	512	0.199296	1.314066
256	256	16	0.020928	97.076456
256	256	256	0.315008	106.311251
256	256	512	1.207808	55.508265
512	512	16	0.023008	353.201667
512	512	256	0.616544	217.268487
512	512	512	2.231648	120.168286
4095	4097	125	--	--

For values of M=4095, N=4097 & K=125, I observed that the GFLOPS was extremely high, and the time was 0.00ms which does not seem correct. I am still working on my code to solve this challenge.