Name: Chukwufumnanya OGBOGU
ID: 011713693

Question 1:
Write cuda code for SpMM where S is represented in CSR format.

Answer:
I wrote the kernel for this question in spmm_csr_driver.cu file attached here. Table 1 is a summary of time and GFLOP/S for different values of K.

K	**small.mtx**		**facebook_combined.mtx**		**2cubes_sphere.asym.mtx**	
	Time (ms)	GFLOPS	Time(ms)	GFLOPS	Time(ms)	GFLOPS
31	0.039296	0.015778	1.152928	4.744883	21.088385	4.842968
32	0.040768	0.015699	1.149088	4.914311	21.509216	4.901382
33	0.041664	0.015841	1.269216	4.588222	22.877855	4.752169
64	0.042368	0.030211	1.854880	6.088778	38.628319	5.458425
128	0.041728	0.061350	3.340288	6.762263	72.912994	5.783600

Question 2:
Write cuda code for SpMM where S is represented in CSC format.

Answer:
I wrote the kernel for this question in spmm_csc_driver.cu file attached here. Table 1 is a summary of time and GFLOP/S for different values of K.

K	**small.mtx**		**facebook_combined.mtx**		**2cubes_sphere.asym.mtx**	
	Time(ms)	GFLOPS	Time(ms)	GFLOPS	Time(ms)	GFLOPS
31	0.042048	0.014745	0.946144	5.781898	19.979008	5.111884
32	0.040736	0.015711	0.979936	5.762597	20.466112	5.151193
33	0.042496	0.015531	1.001088	5.817115	20.990112	5.179554
64	0.043680	0.029304	1.652192	6.835738	36.587486	5.762894
128	0.043232	0.059215	3.015744	7.489994	68.968925	6.114342

Question 3:
Compare CSR VS CSC implementation. What are the advantages and disadvantages of each format for this problem?

Answer:
The CSC implementation achieves slightly higher performance than the CSR implementation. However, CSR matrices are more efficient in accessing row operations, while CSC is more efficient at column operations.

Question 4*:

K	small.mtx		facebook_combined.mtx		2cubes_sphere.asym.mtx	
	Time(ms)	GFLOPS	Time(ms)	GFLOPS	Time(ms)	GFLOPS
64						

Optimizations:
Scatter-Vector SpMM GPU implementation [1]: In this implementation, the threads in a thread block collectively process each row in the sparse matrix (S), and all the thread blocks cycle through all rows in S. Thus, the threads sequentially process the elements in each row. An array is created in global memory to keep track of what each thread is processing thus creating a scatter vector for each warp consisting of threads.

Challenges:
While writing the kernels, here are some of the challenges I encountered;

1) I noticed that using shared memory caused a drop in the performance (GFLOPs) of the kernel as well as a slight correctness issue.

2) I faced some correctness issues when I implemented spmm using cuda streams. Kindly note that this section of my code is commented-out.

References

[1] R. Kunchum, A. Chaudhry, A. Sukumaran-Rajam, Q. Niu, I. Nisa, and P. Sadayappan, “On improving performance of sparse matrix-matrix multiplication on GPUs,” *Proc. Int. Conf. Supercomput.*, vol. Part F1284, 2017, doi: 10.1145/3079079.3079106.

*still working on the Cuda implementation based on the algorithm presented in [1]