


Name: Enrolment No:			
<p style="text-align: center;">UPES End Semester Examination, May 2025</p>			
Course: GPU Programming Program: B. Tech (CSE), Graphics & Gaming Course Code: CSGG3018		Semester : VI Time : 03 hrs. Max. Marks : 100	
Instructions: Please attempt according to the provided time and given weightage.			
SECTION A (5Qx4M=20Marks)			
S. No.		Marks	CO
Q 1	Define the term GPGPUs. List two tools used for GPGPU development.	3+1	CO1
Q 2	Discuss the key difference between task parallelism and data parallelism, providing relevant examples to illustrate each concept.	4	CO1
Q 3	Describe the function of the Thread Execution Manager in GPU architecture and list its primary responsibilities (name each responsibility only, no explanation required).	2+2	CO2
Q 4	Differentiate deadlocks from race conditions, providing effective prevention/detection methods.	4	CO1
Q 5	Differentiate between Concurrency and Parallelism by explaining at least two key differences	4	CO1
SECTION B (4Qx10M= 40 Marks)			
Q 6	List two factors that limit a CUDA kernel from achieving a million times speedup even when the compute-to-global memory access (CGMA) ratio is one. Suggest mitigations for each of these issues.	10	CO3
Q 7	(i) Explain how tiled matrix multiplication improves performance in GPU computing. (ii) Using a 4×4 matrices A and B , illustrate the two-phase computation steps involved in the process of tiled matrix multiplication. Showing (a) how a 2x2 tile of A and B is loaded into shared memory (Phase 1) (b) the step-by-step computation of one output tile in C (Phase 2) using the loaded tiles.	2+8	CO3
Q 8	(i) Given below is a CUDA kernel <pre>global void kernel(int *a) {</pre>		

	<pre>int i = threadIdx.x + blockIdx.x * blockDim.x; if (i % 3 == 0) { a[i] = blockIdx.x * 10 + threadIdx.x; } }</pre> <p>The launch configuration is kernel<<<2, 6>>>(a);</p> <p>If the initial state of a = [-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1] Determine the final state of array a after kernel execution.</p> <p>(ii) For the below CUDA kernel <pre>__global__ void kernel(int *a) { int i = threadIdx.x + blockIdx.x * blockDim.x; a[i] = threadIdx.x % 2; }</pre> The launch configuration is kernel<<<1, 6>>>(a); What is the output array a?</p> <p>(iii) For the below CUDA kernel <pre>__global__ void kernel(int *a) { int i = threadIdx.x + blockIdx.x * blockDim.x; a[i] = i + 1; }</pre> The launch configuration is kernel<<<2, 4>>>(a); What is the output array a?</p>	5 + 2.5 +2.5	CO2, CO3
Q 9	<p>(i) For each CUDA memory types-- registers, shared, global, local, and constant, compare two defining traits such as: scope, performance, lifetime or access constraints.</p> <p style="text-align: center;">OR</p> <p>(ii) Given two matrices, A and B, each of dimension m × m: (a) Write a simple CUDA kernel to compute the product C = A × B. Assume matrices are stored in row-major order. (b) Specify the kernel launch configuration when the matrix dimension is 100 × 100. Justify your choices.</p>	10 7+3	CO2, CO3
SECTION-C (2Qx20M=40 Marks)			
Q 10	<p>(i) Discuss at least two key advantages of OpenACC.</p> <p>(ii) Give two code examples: 1. A loop without data dependencies. 2. A loop with data dependencies. Describe in detail how OpenACC handles each case when #pragma acc parallel loop is applied. Clearly state how the compiler reacts in each case.</p> <p style="text-align: center;">OR</p> <p>(iii) Compare OpenACC's kernels construct ("#pragma acc kernels")</p>	5+15	CO4

	with parallel construct paired with the loop directive (" #pragma acc parallel loop "), providing suitable code examples to illustrate differences in parallelization approach and compiler behavior.	20	
Q 11	<p>(i) Explain the concepts of global dimensions, local work-groups and work-items in OpenCL. Compare these compute model concepts to its CUDA programming equivalents.</p> <p>(ii) Illustrate OpenCL's memory model with a labeled diagram showing: processing elements, compute units, register/global memory, and their access relationships.</p>	12+8	CO3