MANIPAL INSTITUTE OF TECHNOLOGY
BENGALURU
(A constituent unit of MAHE, Manipal)

LMS QUIZ
VI-SEMESTER (CSE Core)
PARALLEL COMPUTER ARCHITECTURE & PROGRAMMING (CSE_3252)

Max Marks :10
Duration: 30-minutes

Student name	Reg no.	Section	Semester

Q NO.	Question	Marks
Q1	Question: Which MPI function ensures that all processes are going into the measured section of code at more or less the same time **A**. MPI_Barrier **B**. MPI_Gather **C**. MPI_SSend **D**. MPI_Scan	**1M**
Q2	You need to implement a fault-tolerant communication system where send operations should not block, even if the receiver is not immediately ready. Design an MPI communication scheme using MPI_Bsend and buffer management. Describe the steps involved, including buffer allocation and potential error handling **A**. Allocate a buffer with MPI_Buffer_attach, use MPI_Bsend for sends, and implement a check to ensure sufficient buffer space before each send. **B**. Allocate a large buffer with MPI_Buffer_attach, use MPI_Bsend for all sends, and ignore potential buffer overflow errors. **C**. Use MPI_Send and implement a timeout mechanism to handle receiver unavailability. **D**. Allocate a buffer with MPI_Buffer_attach, use MPI_Ssend for all sends, and handle errors.	**1M**
Q3	Consider an MPI program that uses MPI_Send and MPI_Recv for pairwise communication between processes. You observe that the program's performance degrades significantly as the number of processes increases. Which of the following factors could contribute to this performance issue? (Select all that apply) **A**. Incorrect usage of MPI_Comm_size **B**. Contention for network resources due to a large number of point-to-point messages. **C**. Inefficient communication patterns leading to excessive idle time. **D**. The use of MPI_Barrier for synchronization.	**1M**
Q4.	Which of the following is true when combining OpenCL and MPI in a hybrid computing environment? **A.** MPI is used for GPU computation, and OpenCL is for inter-process communication. **B.** Both MPI and OpenCL require the same device context initialization. **C.** OpenCL handles task scheduling, while MPI handles memory management. **D.** MPI handles task parallelism, while OpenCL handles data parallelism.	**1M**

Q5	`__kernel void conditional_processing(__global int *data, const int size, const int threshold) {`
	`int gid = get_global_id(0);`
	`int value = data[gid];`
	
	`if (gid < size) {`
	`if (value < threshold) {`
	`data[gid] = value * value; // Square the value`
	`} else if (value == threshold) {`
	`data[gid] = 0; // Set the value to 0`
	`} else {`
	`data[gid] = sqrt((float)value); // Calculate the square root`
	`}`
	`}`
	`}`
	How does this kernel modify the elements of the input array?
	A. It squares all elements.
	B. It sets all elements to 0.
	C. It squares elements less than threshold, sets elements equal to threshold to 0, and calculates the square root of elements greater than threshold.
	D. It calculates the square root of all elements
		1M

Q6	When choosing between using a single command queue with in-order execution or multiple command queues with out-of-order execution, which of the following scenarios would Favor the use of multiple out-of-order command queues? **A.** When strict sequential execution of kernels and data transfers is required. **B.** When the application performs only simple kernel executions. **C.** When minimizing the complexity of synchronization is essential. **D.** When maximizing parallelism and overlapping operations is a priority.	**1M**

Q7	`#include <mpi.h>`
	`#include <stdio.h>`
	`int main(int argc, char *argv[]) {`
	`int rank, size, data;`
	`MPI_Init(&argc, &argv);`
	`MPI_Comm_rank(MPI_COMM_WORLD, &rank);`
	`MPI_Comm_size(MPI_COMM_WORLD, &size);`
	`if (rank == 0) {`
	`data = 100;`
	`MPI_Send(&data, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);`
	`} else if (rank == 1) {`
	`MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);`
	`printf("Process 1 received data: %d\n", data);`
	`}`
	`MPI_Finalize();`
	`return 0;`
		0.5M

	} What will be the output of the program when executed with 2 MPI processes? **A**. Process 1 received data: 100 **B.** Process 1 received data: 1 **C.** Process 1 received data: 0 **D**. Compilation Error	

Q8	#include <mpi.h>
	#include <stdio.h>
	int main(int argc, char** argv) {
	MPI_Init(&argc, &argv);
	int rank, value = 0;
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	
	MPI_Allreduce(&rank, &value, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
	printf("Rank %d, Value: %d\n", rank, value);
	
	MPI_Finalize();
	return 0;
	}
	If this code is run with 3 processes, what will be the output for rank 2?
	A. Rank 2, Value: 0
	B. Rank 2, Value: 3
	C. Rank 2, Value: 6
	D. Rank 2, Value: 1	

Q9	You have an OpenCL application that runs successfully on one platform but fails on another with an "invalid context" error. What are the potential reasons for this discrepancy? (Select all that apply) **A.** The platforms have different device capabilities. **B.** The platforms have different CPU architectures. **C**. The platforms have different OpenCL versions. **D.** The platforms have different operating systems.	
Q10		