Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	BE & Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1701- Distributed Systems				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**05.09.2023**	FN / ~~AN~~
Time: 08:10 - 09:40 A.M (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

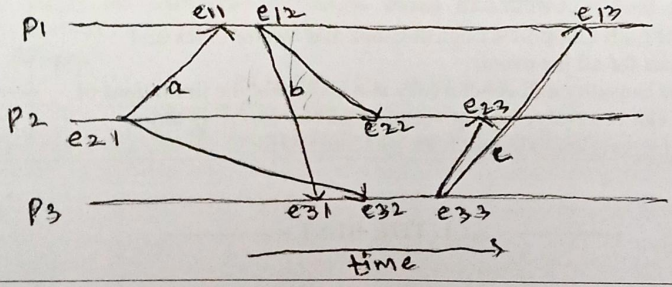
COURSE OUTCOMES

CO1- Realize the foundations of Distributed Systems [K2]
CO2- Able to solve synchronization and state consistency problems [K3]
CO3- Demonstrate the resource sharing techniques in Distributed systems [K3]
CO4- Comprehend the working model of consensus and reliability of Distributed Systems [K3]
CO5- Identify the fundamentals of Peer-to-Peer Systems [K2].
CO6- Formulate a synchronization problem for an ad-hoc distributed system and adapt its solution [K6]

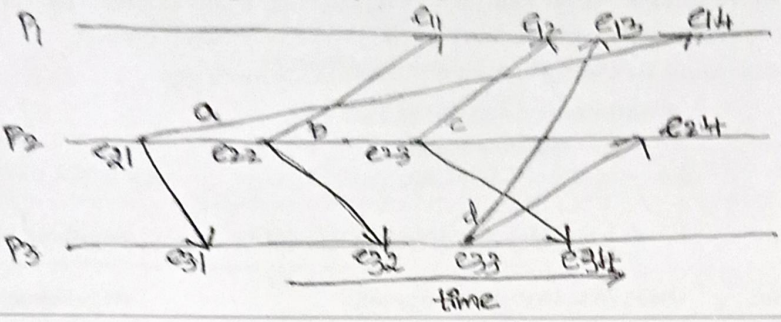
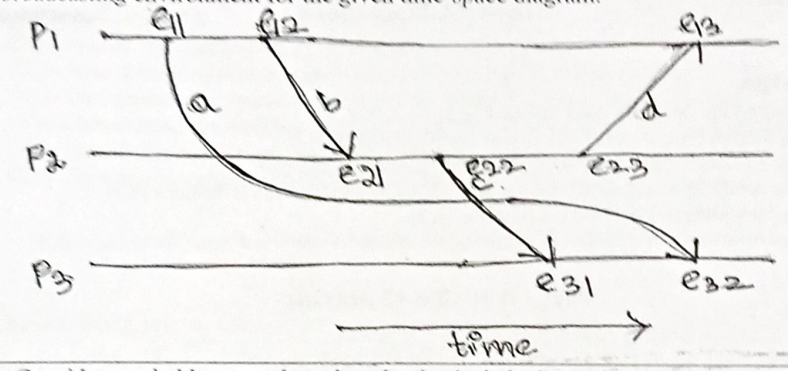
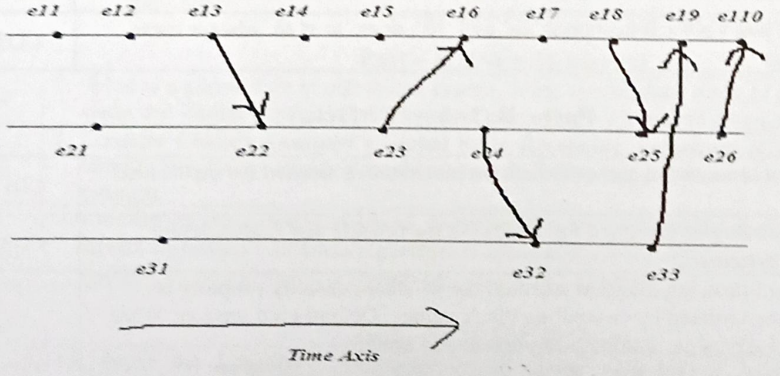
Part – A (6×2 = 12 Marks)

K1	1. List out various architectures for distributed systems based on Flynn's taxonomy.	CO1	3.1.1
K2	2. Classify the various types of transparencies which hides the implementation policies in the distributed systems	CO1	3.1.1
K1	3. List the three forms of load balancing	CO1	2.3.2
K2	4. Discuss and any two major issues in distributed systems	CO2	2.3.2
K3	5. Given two vector timestamps '*ta*' and '*tb*', how do you identify '*ta* < *tb*'	CO2	1.3.1
K3	6. Given two vector timestamps '*ta*' and '*tb*', does '*ta* < *tb*' ***always*** mean '*ta* > *tb*'	CO2	1.3.1

Part – B (3×6 = 18 Marks)

K2	7. Explain synchronization and coordination mechanisms needed for distributed systems.	CO1	2.2.2
K2	8. Outline the strategies designed for achieving the reliable and fault-tolerant distributed systems.	CO1	2.2.2 2.3.2
K4	9. Consider the following diagram and analyze whether causality property is maintained or violated by computing clock values. Defend your answer. What would the clock values computed for event *e13* convey? 	CO2	1.3.1 1.4.1

Part – C (2×10 = 20 Marks)

K3	10. Apply suitable algorithm for the causal ordering of messages in group communication for the given time-space diagram.  	CO2	1.3.1 1.4.1 2.3.1
	(OR)		
K3	11. Apply suitable algorithm for the causal ordering of messages in a non-broadcasting environment for the given time-space diagram.  	CO2	1.3.1 1.4.1 2.3.1
K3	12. Consider a suitable example and apply physical clock synchronization technique for the same. Show that physical clock synchronization is not suitable for Distributed systems through your example. Discuss the effects of clock drift and clock skew.	CO2	1.3.1 1.4.1 2.3.1
	(OR)		
K3	13. Consider the following Time Space diagram and answer the following questions.  i. Apply the rules of both Lamport's Logical Clock and Vector clock and compute the values for all the events. [5] ii. Apply the rule for causality and concurrency and show that the limitations of Lamport's clock can be resolved using vector clocks by choosing any appropriate set of events excluding e11, e12, e21 and e31. [5]	CO2	1.3.1 1.4.1 2.3.1

---------- ALL THE BEST ----------

Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 6 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – II

Question Paper

Degree & Branch	BE & Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1701- Distributed Systems				**Regulation:**	2018
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**11.10.2023**	FN / ~~AN~~
Time: 08:10 - 09:40 A.M (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

COURSE OUTCOMES

CO1- Realize the foundations of Distributed Systems [K2]
CO2- Able to solve synchronization and state consistency problems [K3]
CO3- Demonstrate the resource sharing techniques in Distributed systems [K3]
CO4- Comprehend the working model of consensus and reliability of Distributed Systems [K3]
CO5- Identify the fundamentals of Peer-to-Peer Systems [K2].
CO6- Formulate a synchronization problem for an ad-hoc distributed system and adapt its solution [K6]

Part – A (6×2 = 12 Marks)

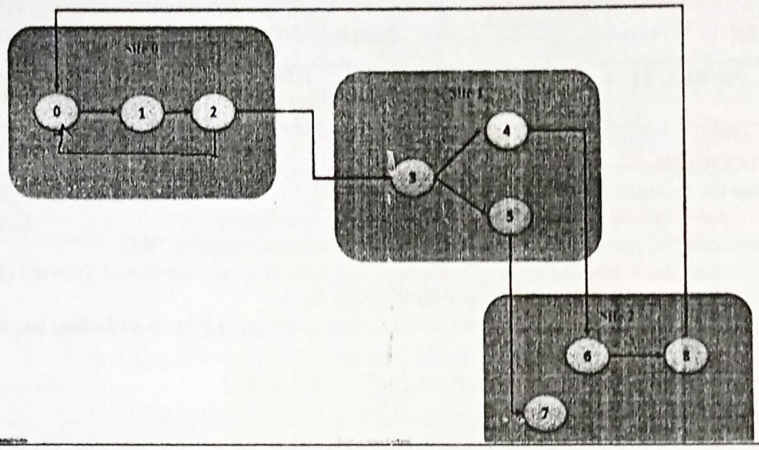
K1	1. Define a Cut in global state of distributed system	CO1	3.1.1
K1	2. What is a transit message in distributed system?	CO2	3.1.1
K1	3. Why does path pushing algorithm detect phantom deadlocks?	CO3	2.3.2
K1	4. What is idle token in mutual exclusion problem?	CO3	2.3.2
K2	5. Outline the Fairness and Liveness properties in Distributed Mutex algorithms.	CO3	2.3.2
K3	6. Identify the number of control messages to be exchanged to achieve one round of mutex in Suzuki Kasami's token based Distributed Mutex algorithm.	CO3	1.4.1

Part – B (3×6 = 18 Marks)

K2	7. What is a global state in distributed system. What are the three types of global states and illustrate three different types of cuts using a space time diagram.	CO1	2.2.3
K3	8. Consider **Chandy Lamport's Global State Recording Algorithm (GSRA).** Identify how GSRA avoids inconsistent snapshot. Illustrate this scenario with an example.	CO2	2.2.2 2.3.2
K4	9. Considering the **non-token based distributed mutex approach, show how Ricart Agrawala's DMutex algorithm is optimal**. Inspect your answer with an example by considering 2 processes in a distributed system.	CO3	1.3.1 1.4.1

Part – C (2×10 = 20 Marks)

K3	10. Apply the **Lamport's non-token based distributed mutual exclusion** algorithm for the scenario in which the order of request for critical section is as follows. **P2 → (P2 || P3) → P1→ P3.** *Note: || Represents concurrent request to CS.* *→ Represents the sequential request to CS.*	CO3	1.3.1 1.4.1 2.3.1

	(OR)		
K3	11. Apply the **token based distributed mutual exclusion algorithm** for the scenario in which the order of request for critical section is as follows. **P1 → (P3 || P2 || P1) → (P1 || P3)** *Note: Initially the token is held by process P3* *|| Represents concurrent request to CS.* *Represents the sequential request to CS.*	CO3	1.3.1 1.4.1 2.3.1
K3	12. Apply **Chandy Misra Haas Algorithm with AND resource model** for the given scenario and identify the presence of deadlocks. Does this algorithm detect any phantom deadlocks? Interpret your answer.  	CO3	1.3.1 1.4.1 2.3.1
	(OR)		
K3	13. Apply **Chandy Misra Haas Algorithm for OR resource model** and demonstrate the process of distributed deadlock detection with an appropriate example.	CO3	1.3.1 1.4.1 2.3.1

---------- **ALL THE BEST** ----------

Register Number | 8 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test - III

Question Paper

Degree & Branch	BE - CSE				**Semester**	7
Subject Code & Name	UCS1701- Distributed Systems				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**03.11.2023**	**FN / ~~AN~~**
Time: 08:10 - 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

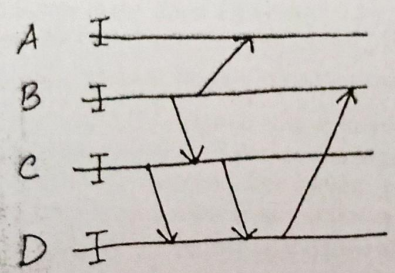
Part – A (6×2 = 12 Marks)

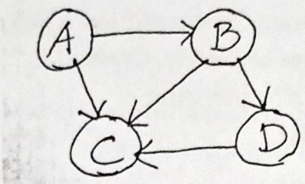
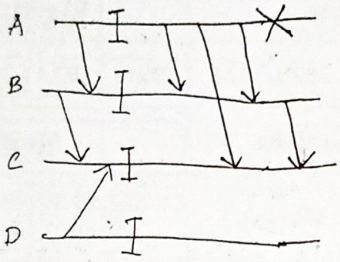
K2	1. If the source is faulty, outline the effect of consensus.	CO4	1.4.1 2.2.3
K1	2. Define Orphan Messages in distributed systems.	CO4	2.2.3
K1	3. What is the difference between a P2P and Client Server system.	CO5	1.4.1 2.2.3
K1	4. Define the term Overlay Networks.	CO	1.4.1 2.2.3
K1	5. Mention any two applications of P2P systems.	CO5	2.1.1
K1	6. What is local indexing and in which type of overlay it is used?	CO5	2.1.1

Part – B (3×6 = 18 Marks)

K3	7. Consider a Distributed System with three processes. Apply **Asynchronous Recovery Algorithm** for the occurrence of a crash at Process 2.	CO4	1.4.1 2.2.3 13.2.1
K2	8. Discuss Distributed indexing of P2P systems in detail.	CO5	1.4.1
K2	9. Explain Lamport's Bakery Algorithm with a suitable example.	CO5	1.4.1 2.1.1

Part – C (2×10 = 20 Marks)

K3	10. Apply **synchronous checkpointing** algorithm for the given scenario in which D initiates the algorithm. 	CO4	1.4.1 2.2.3 13.2.1

	(OR)		
K3	11. Consider the following topology,  Apply **synchronous Recovery** algorithm for the given scenario.  	CO4	1.4.1 2.2.3 13.2.1
K3	12. Apply Byzantine consensus for a distributed System with 4 processes including the source. Discuss the effect of Byzantine Agreement for an Asynchronous distributed systems	CO4	1.4.1 2.2.3 13.2.1
	(OR)		
K3	13. Construct a Live locks scenario for two given processes A & B. Simulate 3 three cycles of rollbacks due to live locks.	CO4	1.4.1 2.2.3 13.2.1

Register Number	2	0	5	0	0	1	0	5	0

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	B.E Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1702- MOBILE COMPUTING				**Regulation**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**06.09.2023**	**FN**
Time: 08:10- 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4 :Analyzing, K5: Evaluating)

Course Outcomes:

CO1	Identify the functionalities of various MAC protocols (K3).
CO2	Explain the functionalities of mobile network layer and routing in Ad hoc networks (K3).
CO3	Analyse the transport and application layer protocols (K3).
CO4	Explain the basics of mobile telecommunication system (K2).
CO5	Develop a mobile application (K3).

Part – A (6×2 = 12 Marks)

K1	1. Why is datalink layer in IEEE 802.11 is subdivided? What are its sublayers?	CO1	2.2.2
K2	2. Compare wired networks and ad-hoc networks.	CO1	2.2.4
K1	3. Why CSMA/CD scheme fails in wireless networks?	CO1	2.2.2
K2	4. Explain Code Division Multiple Access (CDMA).	CO1	2.2.2
K2	5. Outline how packet is delivered from a Mobile Node (MN) to a fixed node / Correspondent Node (CN).	CO2	1.1.2
K1	6. What are the two possibilities for the location of the Care-of Address (COA)?	CO2	2.2.2

Part – B (3×6 = 18 Marks)

K2	7. Illustrate how does DFWMAC- PCF with polling enhance wireless communication in terms of network efficiency and collision reduction.	CO1	1.1.2
K2	8. Compare SDMA, FDMA, TDMA and CDMA.	CO1	2.2.4
K3	9. Imagine a Wi-Fi network with multiple devices contending for access to the wireless channel. A device wants to transmit data, and there are other devices in the network. The spread spectrum followed by the station is frequency hopping. Make use of this spread spectrum and calculate SIFS, PIFS, and DIFS considering the time slot value of 20μs.	CO1	1.4.1 13.3.1

Part – C (2×10 = 20 Marks)

K3	10. Consider a public Wi-Fi in a busy coffee shop. To ensure fair and efficient access to the internet for all users, the Wi-Fi employs a "Multiple Access with Collision Avoidance" mechanism. How does this mechanism function to provide seamless internet connectivity to multiple users simultaneously? Identify the protocols used to prevent data collisions.	CO1	1.1.2 1.4.1 13.3.1
	(OR)		
K3	11. You are the network administrator for a large enterprise. You have a wireless LAN with hundreds of access points and thousands of wireless devices. You are concerned about the high power consumption of the wireless network. Solve the problem of power consumption in your wireless network by implementing a power management solution.	CO1	1.1.2 1.4.1 13.3.1
K3	12. List the entities of mobile IP. Make use of these entities to describe data transfer from a mobile node to a fixed node and vice versa. Why and where encapsulation is needed?	CO2	1.4.1 2.2.4
	(OR)		
K3	13. Identify how tunnelling works in general. Apply this concept for mobile IP and explain how IP-in-IP, minimal, and generic routing encapsulation works. Discuss the advantages and disadvantages of these three methods.	CO2	1.4.1 2.2.4

Register Number | 8 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Computer Science and Engineering

Continuous Assessment Test – II

Question Paper

Degree & Branch	B.E – Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1702- MOBILE COMPUTING				**Regulation**	2018
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	12.10.2023	FN
Time: 08:10- 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4 :Analyzing, K5: Evaluating)

Course Outcomes:

CO1	Identify the functionalities of various MAC protocols (K3).
CO2	Explain the functionalities of mobile network layer and routing in Ad hoc networks (K3).
CO3	Analyse the transport and application layer protocols (K3).
CO4	Explain the basics of mobile telecommunication system (K2).
CO5	Develop a mobile application (K3).

Part – A (6×2 = 12 Marks)

K2	1. Compare proactive and reactive routing protocols.	CO2	2.2.4
K2	2. Explain the concept of "routing overhead" in MANETs, and why is it a concern?	CO2	1.1.2
K2	3. What is the role of a foreign agent in TCP snooping? Outline a foreign agent intercept and analyze TCP traffic between a client and server.	CO3	2.2.2
K2	4. Explain how WDP service is performed.	CO3	1.4.1
K1	5. Draw and show how WTLS establishes a secure session.	CO3	1.1.2
K1	6. Draw and show how WSP/B over WTP - method invocation happens.	CO3	2.2.2

Part – B (3×6 = 18 Marks)

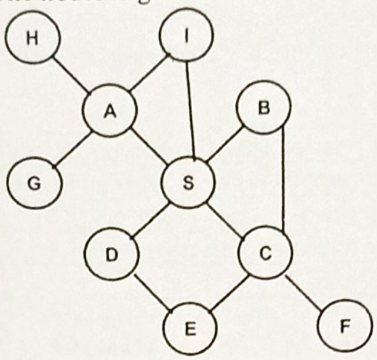
K2	7. Summarize the primary design challenges in MANETs. How do they differ from traditional wired networks?	CO2	2.2.2
K2	8. Compare TCP Tahoe, Reno, and New Reno.	CO3	2.2.4
K3	9. One global goal of Wireless application environment is to minimize over- the-air traffic and resource consumption on the hand-held device. Utilize the WAE logical model to verify this statement.	CO3	2.2.2

Part – C (2×10 = 20 Marks)

K4	Consider the following routing table for mobile node S in an infrastructure less network.	CO2	1.1.2 1.4.1

Destination	Next Hop	Hop Count
A	A	1
B	B	1
C	C	1
D	D	1
E	D	2
F	C	2
G	A	2
H	A	3
I	I	1

The snapshot of mobile nodes is given below.

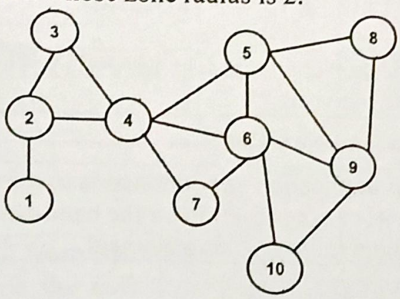


The mobile node S's routing table is updated by a proactive routing protocol after node D is disconnected from the network. Analyze the scenario and rebuild S's updated routing table. [Assume no other nodes move in that time interval]. Also, inspect the scenario where node H needs to communicate to node C regularly, but no other nodes communicate among themselves. How would you modify the routing protocol? Suggest any other considerations which would impact the choice of protocol to use?

(OR)

11. Imagine a group of drones deployed for search and rescue operations in a disaster-stricken area. These drones need to communicate and coordinate their efforts efficiently to locate survivors and report their findings to a central command. Consider the following scenario of drone deployment position whose zone radius is 2:

K4



CO2

Drone 2 is the sender and drone 8 is the receiver. Inspect the scenario and discover the path from 2 to 8. While discovering the path, analyze and write what happens inside and outside the zone of drone 2. Now, assume another scenario in which drone 2 is sender and drones 7,8, 9 are receivers of a group communication. Examine why mobile routing protocol of the above scenario is not suitable. Suggest a more appropriate protocol for group communication, justifying its principles and advantages.

K4	12. Show and explain the working of Indirect-TCP. Draw the packet flow from a fixed host to a mobile host via a foreign agent. List down the actions of the mobile node and the foreign agents during a handover.	CO3	1.4.1 2.2.2
	(OR)		
K4	13. Imagine you are designing a mobile banking application that allows users to check their account balances, make transfers, and pay bills using their smartphones. The application needs to ensure secure and reliable communication between the user's device and the bank's server over a wireless network. You decide to implement the Wireless Transaction Protocol (WTP) to achieve this. Analyze the scenario and describe the transaction model used in WTP. How does it ensure reliable communication in wireless networks, and how can this benefit your mobile banking application?	CO3	1.4.1 2.2.2

Register Number	2	0	5	0	0	1	0	5	0

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Computer Science and Engineering

Continuous Assessment Test – III

Question Paper

Degree & Branch	B.E – Computer Science and Engineering				Semester	VII
Subject Code & Name	UCS1702- MOBILE COMPUTING				Regulation	2018
Academic Year	2023-2024 ODD	Batch	2020-2024	Date	06.11.2023	FN
Time: 08:10- 09:40 AM (90 Minutes)	Answer All Questions				Maximum: 50 Marks	

(K1: Remembering, K2: Understanding, K3: Applying, K4 :Analyzing, K5: Evaluating)

Course Outcomes:

CO1	Identify the functionalities of various MAC protocols (K3).
CO2	Explain the functionalities of mobile network layer and routing in Ad hoc networks (K3).
CO3	Analyse the transport and application layer protocols (K3).
CO4	Explain the basics of mobile telecommunication system (K2).
CO5	Develop a mobile application (K3).

Part – A (6×2 = 12 Marks)

K2	1. Summarize the limitations of GSM.	CO4	1.4.1
K3	2. A GSM subscriber travels abroad. The subscriber needs to make a phone call to his son who lives in his home country. Select a teleservice he would use to make the phone call. Name few other teleservices offered to the subscriber.	CO4	1.1.2
K2	3. Illustrate Mobile originated call.	CO4	2.2.2
K1	4. Define Mobile OS.	CO5	1.4.1
K1	5. List the benefits of using a layered architecture.	CO5	1.4.1
K1	6. What is Android SDK?	CO5	1.4.1

Part – B (3×6 = 18 Marks)

K3	7. A person makes a call from his wired telephone to his friend who uses a mobile phone. Choose the call type of GSM and explain with a neat diagram.	CO4	2.2.2
K3	8. Consider a scenario where you are the chief security officer of a major telecommunications company that operates a GSM network. Your network covers a vast area, providing mobile communication services to millions of users. Security is a top priority to protect user data and the integrity of your network. Identify the security services provided by GSM and illustrate these services with a suitable diagram.	CO4	2.2.4

K2	9. Summarize the role and significance of the Linux kernel in the Android architecture. How does it interact with higher-level Android components?	CO5	2.2.4

Part – C (2×10 = 20 Marks)

K2	10. What are the different components of the Global System for Mobile Communications (GSM) architecture? Explain the components with a neat diagram.	CO4	2.2.2
	(OR)		
K2	11. Describe the architecture and components of GPRS with a neat diagram.	CO4	2.2.2
K3	12. An iOS developer develops an application that allows users to take and share photos. Your application needs to be able to access the camera to take photos, access the photo library to save and share photos, and access the network to share photos with other users. Identify the layers of iOS Architecture involved in developing this application and explain them with a neat diagram.	CO5	1.4.1
	(OR)		
K3	13. A team develops a multifaceted Android application. Make use of the primary building blocks of Android to develop a multifaceted application. Explain how these building blocks will be utilized to create a feature-rich user experience Android application.	CO5	1.4.1

Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	B.E CSE				**Semester**	VII
Subject Code & Name	UCS1703 Graphics and Multimedia				**Regulation:**	2018
Academic Year	2023-2024 ODD	**Batch**	2020-24	**Date**	07.09.2023	FN
Time: 08:10 - 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1	Apply the algorithms to manipulate output primitives such as line, circle, ellipse (K3)
CO2	Demonstrate transformations, representations and clipping on 2D objects and map window to viewport transformations (K3)
CO3	Apply three Dimensional concepts like representations, geometric transformations, and projections (K3)
CO4	Understand the working of different illumination and color models used to render an animation scene (K2)
CO5	Understand different types of multimedia file formats, compression techniques and design basic 3D Scenes using Blender (K2)

Part – A (6×2 = 12 Marks)

K1	1. In which component of the graphics system is the picture definition stored and how?	CO1	1.4.1 13.2.1
K1	2. What is persistence in a CRT monitor? How does it affect the refresh rate?	CO1	1.3.1 1.4.1
K3	3. DDA line drawing algorithm is applied to plot a line AB from A(1,6) to B (5,9). Find (x_2, y_2) when (x_1, y_1) = (2,7).	CO1	1.1.1 1.4.1 2.1.3 13.3.1
K2	4. Illustrate the 2D graphics pipeline.	CO2	1.4.1
K3	5. Identify the correct answer. Shearing is a transformation that a. distorts but does not move the object b. distorts and may move the object based on the shearing reference line c. moves the object but does not distort the object d. retains the original shape of the object but enlarges or decreases the size Justify your answer by applying shearing on a unit square.	CO2	1.4.1 2.1.3
K2	6. Compare and contrast Uniform and Differential scaling.	CO2	1.1.1 1.4.1 2.1.2 13.2.1

Part – B (3×6 = 18 Marks)

K2	7. Explain random and raster scan systems. Show using diagrams how basic output primitives are drawn in these systems.	CO1	1.3.1 1.4.1 13.2.1
K3	8. Apply Midpoint circle drawing algorithm to plot the points on the circle with centre = (4,5) and radius = 4. Show how each point (x_k, y_k) is calculated.	CO1	1.1.1 1.4.1 2.1.3 13.3.1
K2	9. Illustrate the technique of interlacing with suitable diagrams.	CO1	1.3.1 1.4.1

Part – C (2×10 = 20 Marks)

K3	10. Apply the Bresenham's line drawing algorithm for the given points (2,1) and (10,12). Plot the pixels and draw the line. Show how each point (x_k, y_k) is calculated.	CO1	1.1.1 1.4.1 2.1.3 2.4.1 13.3.1
	(OR)		
K3	11. Apply the ellipse drawing algorithm to plot points on the ellipse with $r_x = 9$ and $r_y = 7$. Show how each point (x_k, y_k) is calculated.	CO1	1.1.1 1.4.1 2.1.3 2.4.1 13.3.1
K3	12. a. Rotate a triangle (4,6), (2,2), (6,2) about the vertex (4,6) by 180 degrees counterclockwise and find the new vertex. Use homogeneous co-ordinate representation.	CO2	1.1.1 1.4.1 2.1.3 2.4.1 13.3.1
	(OR)		
K3	13. Apply reflection on a square with vertices (2,2) (6,2) (2,6) and (6,6) with respect to the line x = y and with reflection factor = 2. Use homogeneous co-ordinate representation.	CO2	1.1.1 1.4.1 2.1.3 2.4.1 13.3.1

Register Number | 8 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – II

Question Paper

Degree & Branch	B.E CSE				**Semester**	VII
Subject Code & Name	UCS1703 Graphics and Multimedia				**Regulation:**	2018
Academic Year	2023-2024 ODD	**Batch**	2020-24	**Date**	13.10.2023	FN
Time: 08:10 - 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

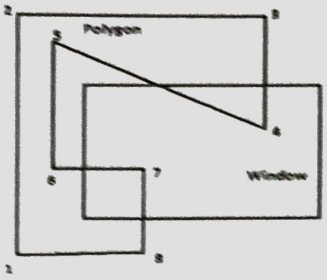
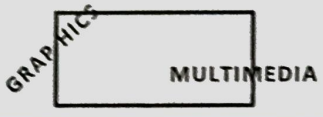
(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1	Apply the algorithms to manipulate output primitives such as line, circle, ellipse (K3)
CO2	Demonstrate transformations, representations and clipping on 2D objects and map window to viewport transformations (K3)
CO3	Apply three Dimensional concepts like representations, geometric transformations, and projections (K3)
CO4	Understand the working of different illumination and color models used to render an animation scene (K2)
CO5	Understand different types of multimedia file formats, compression techniques and design basic 3D Scenes using Blender (K2)

Part – A (6×2 = 12 Marks)

K1	1. What is exterior clipping? Give an example scenario.	CO2	1.4.1
K2	2. Explain the four cases of clipping a line segment PQ of a polygon using Sutherland-Hodgeman Polygon clipping algorithm.	CO2	2.1.3
K2	3. Illustrate the calculation of the transformed point *(x',y')* in a viewport with co-ordinates *XVmin, XVmax, YVmin, YVmax* given *(x,y)* in the window with co-ordinates *XWmin, XWmax, YWmin* and *YWmax*.	CO2	1.4.1
K3	4. Identify the type of the following 3D object and write its characteristics. 	CO3	1.4.1
K1	A point *P(x,y,z)* is translated using a translation vector *T(a,b,c)*. Write the transformation matrix to find the transformed point *P'*.	CO3	1.1.1 1.4.1
K1	6. List any two three-dimensional display methods and their applications.	CO3	1.4.1

Part – B (3×6 = 18 Marks)

K3	7. Apply Weiler-Atherton polygon clipping algorithm for the following example. 	CO2	1.4.1 2.1.3 13.3.1
K3	8. Apply all three types of text clipping for the strings shown in the figure below. 	CO2	1.4.1 2.1.3 13.2.1
K2	9. Explain in detail how an object is reflected in 3D space. Draw suitable diagrams.	CO3	1.3.1 1.4.1 13.3.1

Part – C (2×10 = 20 Marks)

K3	10. A cube lies in a 3D space with *x,y,z* as the principal axis in a right handed co-ordinate system. An axis *A* passes through points ***P(x1,y1,z1)*** and ***Q(x2,y2,z2)*** and is not parallel to any of the principal axes. Apply rotation on the cube by an angle θ about the axis *A*. Illustrate the steps and matrices of the transformation.	CO3	1.4.1 2.4.1 13.3.1
	(OR)		
K3	11. Apply the composite transformation for reducing a unit cube to half its volume by keeping the centroid fixed. Use only homogeneous coordinate representation.	CO3	1.4.1 2.4.1 13.3.1
K3	12. Apply Cohen Sutherland clipping algorithm and clip the line ***A (5,120) B (50,80)*** with respect to the window with ***Xleft=10, Xright=60, YBot=10, Ytop=100.***	CO2	1.4.1 2.1.3 13.3.1
	(OR)		
K3	13. Apply the Liang-Barsky algorithm to clip the line ***P1(40,15) P2(75,45)*** with respect to the window with ***(XWmin,YWmin) = (50,10) and (XWmax,YWmax) = (80,40).***	CO2	1.4.1 2.1.3 13.3.1

Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – III

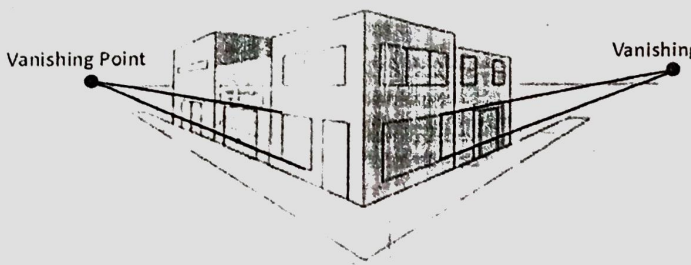
Question Paper

Degree & Branch	B.E CSE				**Semester**	VII
Subject Code & Name	UCS1703 Graphics and Multimedia				**Regulation:**	2018
Academic Year	2023-2024 ODD	**Batch**	2020-24	**Date**	07.11.2023	FN
Time: 08:10 - 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1	Apply the algorithms to manipulate output primitives such as line, circle, ellipse (K3)
CO2	Demonstrate transformations, representations and clipping on 2D objects and map window to viewport transformations (K3)
CO3	Apply three Dimensional concepts like representations, geometric transformations, and projections (K3)
CO4	Understand the working of different illumination and color models used to render an animation scene (K2)
CO5	Understand different types of multimedia file formats, compression techniques and design basic 3D Scenes using Blender (K2)

Part – A (6×2 = 12 Marks)

K1	1. What are the three types of Axonometric projections?	CO3	1.4.1
K2	2. Compare and contrast Cavalier and Cabinet projections.	CO3	1.4.1
K3	3. Identify the type of projection shown in the figure below and draw the position of the view plane with respect to the co-ordinate origin to achieve this projection. 	CO3	1.3.1 1.4.1
K1	4. What is ambient light and what is the contribution of ambient light in adding visual realism to a scene?	CO4	1.2.1

K3	5. Keyframe *k* shows a racecar parked in its initial position. Keyframe *k+n* shows the car participating in the race. Identify how the in-between frames from keyframe *k* to keyframe *k+n* are placed on a timeline graph to show the acceleration of the car.	CO4	1.1.1 2.1.2
K1	6. List two modern multimedia tools for rendering 2D and 3D graphics.	CO5	5.1.1

Part – B (3×6 = 18 Marks)

K2	7. Explain the steps in the design of an animation sequence.	CO4	13.3.1
K3	8. A scene is rendered where the objects appear shiny and have highlights. Identify and derive the component of the illumination model used to render the scene when the vector between the light source and the object is given as *s* and vector between the object and the viewer is given as *v*.	CO4	2.1.2 13.2.1
K3	9. A color C in a three primary color model is defined with components (c1, c2, c3). Identify the representation of color C in additive and subtractive color systems and explain it with one example for each system.	CO4	2.1.2 13.2.1

Part – C (2×10 = 20 Marks)

K3	10. Consider a 3D coordinate system where y-axis is vertical, and z-axis is pointing towards the viewer. A line with endpoints A(15,10,5) and B(10,15,20) is projected onto XY plane. Apply Cavalier projection with $\varphi = 45$ degrees and calculate the projected line endpoints.	CO3	1.4.1 13.3.1
	(OR)		
K3	11. A point (x,y,z)=(3,2,1) is projected using oblique parallel projection to a position (x_p,y_p) on the XY plane. The projector makes an angle $\alpha = 45$ with the line of length L on the projection plane that joins (x_p,y_p) and (x,y). The line L makes an angle $\phi = 30$ with the horizontal direction in the projection plane. Apply the projection to find the point (x_p,y_p).	CO3	1.4.1 13.3.1
K2	12. Explain in detail the JPEG Compression scheme.	CO5	1.4.1
	(OR)		
K2	13. Explain with a neat diagram the architecture of a multimedia system and its defining objects.	CO5	1.4.1

Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	B.E. Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1704 – Management and Ethical Practices				**Regulation:**	2018
Academic Year	2023 – 2024 (ODD)	**Batch**	2020-2024	**Date**	08.09.2023	FN
Time: 08:10 – 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

Part – A (6×2 = 12 Marks)

K1	1. What are the essential skills of Managers?	CO1	10.2.1
K1	2. List down the types of departmentalization.	CO2	10.1.3
K2	3. Distinguish between plan and goal.	CO1	10.2.1
K1	4. List all the decision-making biases and errors.	CO1	10.1.3
K1	5. Specify the functions of management.	CO1	10.2.1
K1	6. Define Organization.	CO1	10.1.3

Part – B (3×6 = 18 Marks)

K2	7. General Administrative Theories of Henry Fayol and Max Weber give contrasting ideas. Discuss the key features of each.	CO1	8.1.1
K2	8. Explain the classification of managers in detail.	CO1	6.1.1
K3	9. At the Emil Sporting Goods factory in Canada, 150 workers make football used in the National Football League, college and high school football games. In order to achieve high productivity, the workers specialize in different tasks such as molding, stitching and sewing, lacing, and so forth. Beyond a point, the management observed that productivity started to degrade. Using a suitable graph, illustrate the situation that the management has observed and compare it with their performance earlier.	CO2	6.1.1

Part – C (2×10 = 20 Marks)

K3	10. Compare and contrast quantitative approach and contingency approach for international business. NMK is a start-up company that designs mini robots for water quality assessment. Derive your inferences on both approaches when applied for marketing these mini-robots.	CO1	1.3.1, 10.2.1

	(OR)		
K3	11. Distinguish a strong cultured organization from a weak cultured organization. Las Vegas Productions is a production company. In order to establish and maintain it as a strong cultured organization right from its launching, identify the major steps involved.	CO1	1.3.1, 10.2.1
K3	12. Consider the following scenarios: **Scenario 1:** Kumar is a mid-level manager. He keeps all his subordinates under a lot of discipline. His employees complain about the waste of time and effort, as they feel that nothing is being assigned in a proper way and a proper place, also no proper schedule is made for working. **Scenario 2:** ABC is a company which takes care of the fact that the confidence of the employees should always be at its peak. For this reason, they give assurance to their employees for employment for a minimum fixed tenure of time. a. Identify which principle of management is violated in scenario 1. Infer from the scenario and suggest a suitable managerial solution to overcome it. b. Identify which principle of management is followed in scenario 2. How this principle supports your claim?	CO1	12.1.1
	(OR)		
K3	13. Consider the following scenarios: **Scenario 1:** Ram is the owner of a printing press. The size of his organization has increased during the recent past. There are many employees who work in his organization. The organization is considered good and has earned a lot of reputation in the market. However, when it comes to making key decisions in the organization, he never considers the opinions of his subordinates. Even though the size of the organization has increased he tries to take all the key decisions on his own. **Scenario 2:** Pritam Vehicles is a vehicle manufacturing company. The company has the same unit producing both lorries and vans. This leads to confusion among the employees regarding the reporting as well as differentiation of work. a. Identify the managerial approach he is following in scenario 1. Give a suggestion in order to overcome this issue with suitable justification. b. Identify which principle of Fayol is violated in scenario 2. Why? State the principle. Give an immediate outcome of the violation of this principle.	CO1	12.1.1

Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – II

Question Paper

Degree & Branch	B.E. Computer Science and Engineering				**Semester**	VII
Subject Code & Name	UCS1704 – Management and Ethical Practices				**Regulation:**	**2018**
Academic Year	2023 – 2024 (ODD)	**Batch**	2020 - 2024	**Date**	**16/10/2023**	**FN**
Time: 08:10 – 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Describe basic and applied fields of Management (K2)
CO2:	Describe and practice Managerial skills (K3)
CO3:	Describe and practice Engineering Ethics and Human Values (K3)
CO4:	Describe and use safety, responsibility, and rights (K3)
CO5:	Describe ethical issues in cybersecurity (K2)

Part – A (6×2 = 12 Marks)

K2	1. Explain the need for motivation in an organization	CO2	10.1.2
K2	2. Elaborate the various senses of engineering ethics.	CO3	10.1.3, 8.2.2
K2	3. Illustrate the new organizational configurations.	CO2	10.1.2
K2	4. Compare and contrast consensus and controversy.	CO3	10.1.3, 8.2.2
K1	5. Which are the differing views of professionals?	CO3	12.1.1
K2	6. Differentiate moral dilemma and moral autonomy.	CO3	12.1.1

Part – B (3×6 = 18 Marks)

K2	7. Outline the various senses of responsibility.	CO3	10.1.3
K2	8. Give the pictographic representation and highlight the Kohlberg and Gilligan theories of moral development. Summarize the criticisms for both theories.	CO3	8.2.2
K3	9. Interpret the following scenario and give your opinion to prevent this from happening. A doctor terminated a female staff member who had been working for him for nine months. She then claimed that her civil rights were violated and that she was discriminated because of her age. She was 46 years old. During an investigation, the doctor was asked to provide copies of her performance evaluations to justify the contention that the termination was based	CO3	10.1.3, 8.2.2

	on her performance, not her age. However, no performance evaluations had been done to any employees. This claim led to an out-of-court settlement. Considerations The employee had been working with the doctor for nine months and had not received a performance appraisal. She claimed that she was not aware that her performance was not satisfactory and was surprised when she was terminated. She alleged that the sole reason for her termination was her age — that the doctor wanted to hire a younger person. The doctor said she had been warned on several occasions, but nothing had been documented.		

Part – C (2×10 = 20 Marks)

K4	10. Having a car to get to work is a necessity for many workers. When two crucial employees of Vurv Technology in Jacksonville, Florida, had trouble getting to work, owner Derek Mercer decided to buy two inexpensive used cars for the employees. He said, "I felt that they were good employees and a valuable asset to the company." One of the employees who got one of the cars said, "It wasn't the nicest car. It wasn't the prettiest car. But it is an overwhelming feeling of worry replaced by enlightenment. The 80-hour weeks we worked after that never meant anything. It was "give and take". I was giving and the company was definitely giving back. a. Examine what motivated the employees. b. Infer which motivational theory well suits the above mentioned scenario.	CO2	10.1
	(OR)		
K4	11. You are appointed as a Human Resource Manager to Qubitnew Company. It is a well-established organization. Design a suitable HR Management Process with a suitable diagram for recruiting new employees in the company.	CO2	10.1.2
K4	12. Smith and Tom work in an experimental testing laboratory for Acme Corp. Smith has been the main testing engineer for five years and is up for promotion to laboratory supervisor (includes the testing laboratory and several other laboratories). Tom is being trained as a potential replacement as the testing engineer. The laboratories division supervisor is Brown who is retiring soon. The company's latest development project is an OEM control module for a well water pump. The pump manufacturer has promised an important contract if the module meets their approval. The original module prototype met the desired specifications with the exception of the temperature test. The prototype was sent back to the development engineers for rework. The next iteration of the module was sent to the testing laboratory for testing, but the temperature test was delayed since the needed equipment was out for recalibration. Tom wrote the report for the original prototype and the draft report for the reworked prototype. At the weekly laboratory's division meeting, Smith reports to Brown that the latest module "meets all tests" and that the report has "good number of " equipments for temperature test. Tom questions Smith privately after the meeting since his draft report indicated that temperature testing was delayed. Smith says that the development team fix should be satisfactory, i.e. it was confirmed through simulation, and that they can do further testing later to confirm the simulation numbers once the needed testing equipment is returned. Smith says that a positive report to the manufacturer cannot be delayed or their testing laboratory "looks bad" and the contract could be at risk. Then, Tom privately speaks to Brown about the situation including his concern that his (Tom) name is on the overall	CO3	10.1.3, 6.1.6, 8.2.2

	testing report. Brown tells him that Smith is responsible for the details in the report and that he (Tom) should learn to work with Smith if he wants to take over the testing laboratory. a. Examine the ethical questions in this situation. b. Analyze the scenario and give your insights about Tom's behaviour. c. After the private conversations with Smith and Brown, examine whether Tom has an ethical responsibility to speak to Smith again or not?		
	(OR)		
K4	13. Bart Matthews, a robot operator at Cybernetics, Inc., has been killed by an out-of-control robot named Robbie. The creator of the robot, Silicon Technologies, is also in a tight financial position and had hoped that the robot would put the company back on its feet. It has been determined that several situations contributed to the death of Matthews: • Improper methodology was used in developing the software. • Testing of the software was faked. • The company pressured Robbie's creators to bypass testing. • Part of the software used in the robot was stolen from another vendor's application. • The programmer did not understand or know the code which he used. • Security measures used were illegal, and therefore all information gathered in regard to the case might not be permissible in court. • The project leader did not understand or use proper design methodologies. • The end-user interface was designed improperly. a. With suitable justification, inspect who is at fault. b. Analyze the situations that are unethical. c. From the given situation, give your inference regarding the major contributor/s to the death of Bart Matthews.	CO3	10.1.3, 6.1.6, 8.2.2

K

K2

K3

Register Number 3 1 E C C 1 5 F 0

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

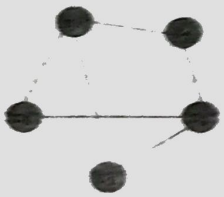
Department of Computer Science and Engineering

Continuous Assessment Test – I

Question Paper

Degree & Branch	B.E., Computer Science & Engineering			**Semester**	V
Subject Code & Name	UCS1722 – Social Network Analysis			**Regulation:**	2018
Academic Year	2023-2024 ODD	**Batch**	2020 - 2024	**Date**	11.09.2023 FN
Time: 08:15 - 09:45 a.m (90 Minutes)	**Answer All Questions**			**Maximum: 50 Marks**	

Part – A (6×2 = 12 Marks)

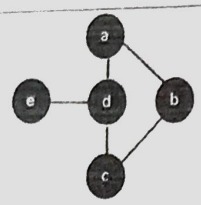
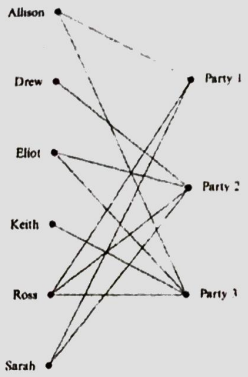
K1	1. What is Social Network Analysis?	CO1	1.4.1
K1	2. What are the different kinds of global structures of social networks?	CO1	1.4.1
K3	3. Apply the social network concepts to find the degree centrality of the nodes in the given network. 	CO1	2.3.1
K3	4. Assume a network of 5 countries that shares common borders such that each country shares common borders with all other countries. Identify whether the network is a directed or undirected and find out the maximum possible borders that can exist between the countries.	CO1	2.2.3
K3	5. Identify a real-world example of an actor who might be powerful but not central? Who might be central, but not powerful?	CO2	2.2.3
K1	6. What are the different notational schemes to represent social network mathematically?	CO2	1.4.1

Part – B (3×6 = 18 Marks)

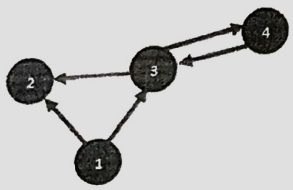
K2	7. Explain the different graph models for social networks and their pros and cons	CO1	1.4.1
K3	8. Apply the social network concepts to find the betweenness centrality and closeness centrality measures for all the nodes in the network given below.	CO1	2.4.1

Register Number 2 0 5 0 0 1 0 5 0

Sri Sivasubramaniya Nadar College of Engineering

			
K3	9. Build one-mode affiliation network for the given two-mode affiliation network, where the actors are children, and the events are the birthday parties they attended. 	CO1	2.4.1

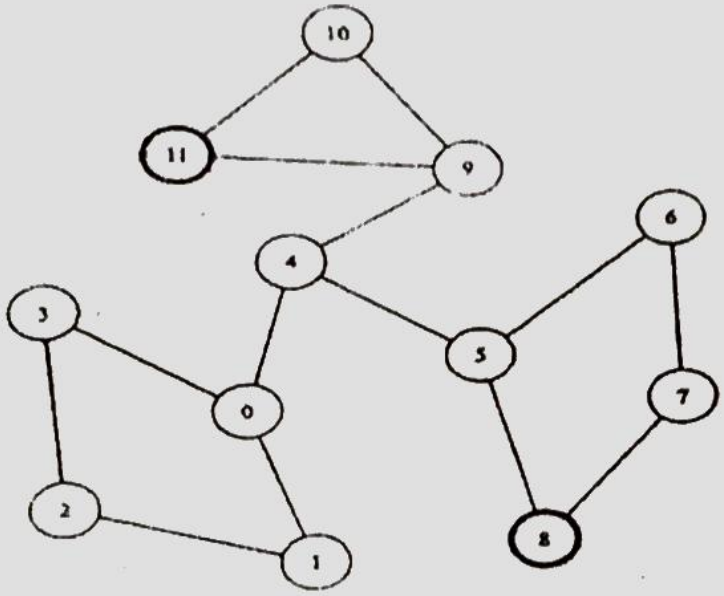
Part – C (2×10 = 20 Marks)

K2	10. Explain the different dimensions of Social Capital by Nahapiet and Ghoshal.	CO1	1.4.1
	(OR)		
K2	11.Explain the data collection complexities in social networks and detail the alternatives.	CO1	1.4.1
K3	12. Apply the appropriate matrix operations for the network shown below and find the following measures.  a) Distance/connectivity matrix b) Geodesic of length 2 c) Indegrees of nodes d) Outdegrees of nodes e) Density of the network	CO2	2.4.1

13. Apply the graph theory concepts for social networks and find the following measures for the network shown below.

a) Eccentricity of Node3
b) A Walk
c) A Trial
d) Tour in the graph
e) Nodal degree of all nodes
f) Mean nodal degree
g) Variance of degree
h) Density of the graph
i) 9.Cut-Point or node connectivity
j) Edge connectivity

CO2



Register Number | 2 | 0 | 5 | 0 | 0 | 1 | 0 | 5 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Department of Computer Science and Engineering

Continuous Assessment Test – II

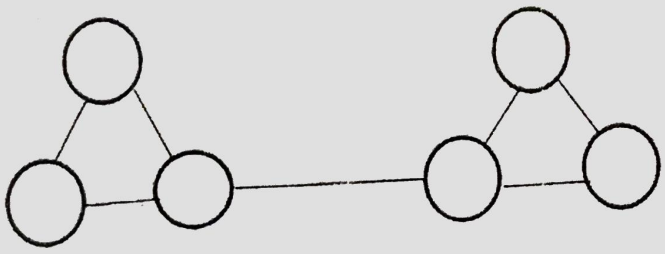
Question Paper

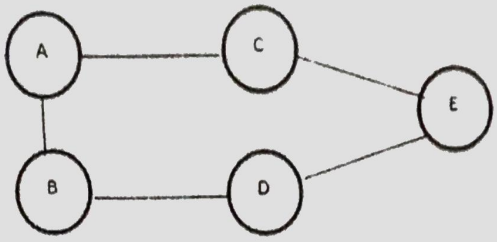
Degree & Branch	B.E., Computer Science & Engineering				**Semester**	7
Subject Code & Name	UCS1722 – Social Network Analysis				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020 - 2024	**Date**	**17.10.23**	**FN**
Time: 08:10 - 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

Part – A (6×2 = 12 Marks)

K1	1. List any four examples of non-directional relations in social network.	CO2	1.4.1
K3	2. Identify the relations for the scenario "children playing together".	CO2	2.2.3
K1	3.Why is detecting communities in a Social Network important?	CO3	1.4.1
K1	4. What is the rewritten form of the formula for the modularity of Newman and Girivan and write down what each term refers to.	CO3	2.2.3
K2	5. Differentiate between optimization-based and heuristics-based algorithms for solving Network Community Mining Problem in Social Networks.	CO3	2.2.3
K3	6. Develop a *NetworkX* code to remove nodes with degree > 2 from the network.	CO4	2.2.3

Part – B (3×6 = 18 Marks)

K2	7. Explain the three categories of definitions for communities.	CO3	1.4.1
K3	8. Apply Newman-Girvan modularity to find the goodness of the partition for the network given below. 	CO3	2.4.2

K3	9. Apply Girvan-Newman's algorithm to find the community in the below network.  	CO4	2 4 2

Part – C (2×10 = 20 Marks)

K2	10. Explain the various optimization-based algorithms for detecting the communities in social networks.	CO3	1 4 1
	(OR)		
K2	11. Explain any three applications of community mining algorithms.	CO3	1 4 1
K4	12. Assume there is a large community-based network. A researcher wants to analyze the network based on certain criteria of interest. But he is clueless where to start. What are the different types of approaches he should follow to simplify the network for analysis. Write the *NetworkX* code for any one of the approaches. Also, suggest different ways to find the most influential node in the network.	CO4	2 4 1
	(OR)		
K4	13. Assume an API that retrieves the connection information between a particular user and his friends circle at several levels in a social network. The connections are directional and are in the following format. Alice > bob Dave < Alice Carol > Alice Write a *NetworkX* code to draw an Ego network for "Alice" at friend-of friend-of-friend degree level. Also, for the same network, write a *NetworkX* code to perform data collection using snowball sampling.	CO4	2 4 1

Register Number | 2 | 6 | 6 | 0 | 0 | 1 | 0 | 6 | 0 |

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110

(An Autonomous Institution, Affiliated to Anna University, Chennai)

Computer Science and Engineering

Continuous Assessment Test – III

Question Paper

Degree & Branch	B.E. Computer Science and Engineering				**Semester**	7
Subject Code & Name	UCS 1722 Social Network Analysis				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020 - 2024	**Date**	**17.11.22**	**FN**
Time: 08:10 – 09:40 a.m (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

Part – A (6×2 = 12 Marks)

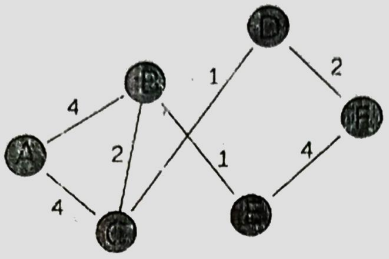
KL1	1. What is meant by triadic analysis?	CO4
KL3	2. Identify the significance of the forbidden triads in a social network.	CO4
KL2	3. How does visualization help in social network analysis?	CO5
KL1	4. Which kind of network is suited for matrix-based visualization?	CO5
KL3	5. Compare and Contrast the functionalities of the 3 most popular centrality measures used in social network analysis.	CO5
KL1	6. List the advantages of Node-Link Diagrams.	CO5

Part – B (3×6 = 18 Marks)

KL2	7. Explain the various kinds of Node-Edge diagrams used for social network visualizations	CO5
KL3	8. Identify the similarities and dissimilarities in the functionalities of user-centric, content-centric and hybrid visualization.	CO5
KL4	9. Distinguish between matrix based and node-link based diagrams.	CO5

Part – C (2×10 = 20 Marks)

KL2	10. Discuss the different solutions that could be used to make visualization in readable form when scaling to larger networks.	CO5
	(OR)	
KL2	11. Explain how the combination of matrix and node link diagrams is leveraged for better visualization.	CO5
KL3	12. Develop a *NetworkX* code to draw a Krackhardt kite graph with 7 nodes and display the following: a) adjacency list b) neighbours of each node in the graph c) depth first traversal with node 0 as starting point d) Shortest path from node 0 to node 7	CO5

	e) Average shortest path of the graph	
	(OR)	
KL3	13 a. Apply Node-link based visualization technique to draw the below network using *NetworkX* code as a labelled weighted graph. Node's colors should be in green and the link's colors should be in blue. Use spring layout for positioning. b. Write an algorithm to perform an appropriate traversal to analyze a person's network connection at a deeper level. Using *NetworkX* command implement the traversal method and display the list of nodes in the order it was visited. 	CO5

17-

4840

1 & 15

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -1** **Question Paper**						
Degree & Branch	B.E				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**12.09.2023**	**FN**
Time: 08:10 - 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	K1	What are Warps? What is the size of a Warp defined by nVidia?	CO1	1.3.1 2.2.2 13.1.1
2	K1	What is the bandwidth of PCI-E lanes in Core-2 series GPU architecture?	CO1	1.3.1 1.4.1 2.2.2
3	K2	What are PTX instructions? Explain the fields of the PTX instruction format.	CO1	1.3.1 2.1.2
4	K1	List the differences between CPU and GPU	CO1	1.3.1 2.2.2
5	K1	How does thread divergence impact the performance of CUDA applications?	CO1	1.3.1 2.1.2 2.2.2 13.1.1 13.3.1
6	K3	Identify the CUDA API call that make sure that all previous kernel executions and memory copies have been completed.	CO2	1.4.1 2.1.2 13.3.1

Part – B (3×6 = 18 Marks)

7	K3	1) It is required to use each thread to calculate two(adjacent) elements of vector addition. Assume that variable **i** is the index of the first element to be processed by a thread. Identify the correct expression for mapping the thread/block indices to data index? Justify your Response..	CO1	1.3.1 1.4.1 2.2.2 13.1.2 13.3.1
8	K2	Explain the Memory Structure of GPU processor.	CO1	1.3.1 1.4.1 2.2.2 13.1.1
9	K3	Assume that the GPU is processing vector **A** that has 64 million data. Assume that one thread is launched for each element of a vector. If they group 1024 elements in a block and assign one block to each streaming multiprocessor, a)Estimate the number of threads, warps, blocks needed to process the data. b)Estimate the number of streaming multiprocessors needed in the system. c) Estimate the storage space requirements for storing: i)single precision floating point data. ii) Double precision floating point data	CO1	1.3.1 1.4.1 **2.2.2** 13.3.1

Part – C (2×10 = 20 Marks)

10	K2	Explain with a neat block diagram, the architecture of Core 2 series GPU processor.	CO1	1.3.1 1.4.1 2.2.2
		(OR)		
11	K2	Explain various CUDA compute levels in detail	CO1	1.3.1 1.4.1 2.2.2
12	K3	Explain the process of performing vector addition using CUDA programming. Provide a step-by-step explanation of how to parallelize the addition of two large vectors on a GPU, Discuss the advantages of using CUDA for vector addition compared to a sequential CPU approach. Develop a CUDA program to perform addition of two vectors.	CO2	1.4.1 2.1.2 13.3.1 13.3.2
		(OR)		
13	K3	Describe the key steps involved in implementing Radix Sort in GPU systems, including how you would handle parallelization and memory management Develop a CUDA program to sort list of elements in an array using Radix sort.	CO2	1.4.1 2.1.2 13.3.1 13.3.2

Prepared By **PAC Members** **Approved By**

(HOD/CSE)

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -1** **Question Paper**						
Degree & Branch	B.E				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**.09.2023**	**FN**
Time: 08:15 - 09:45 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

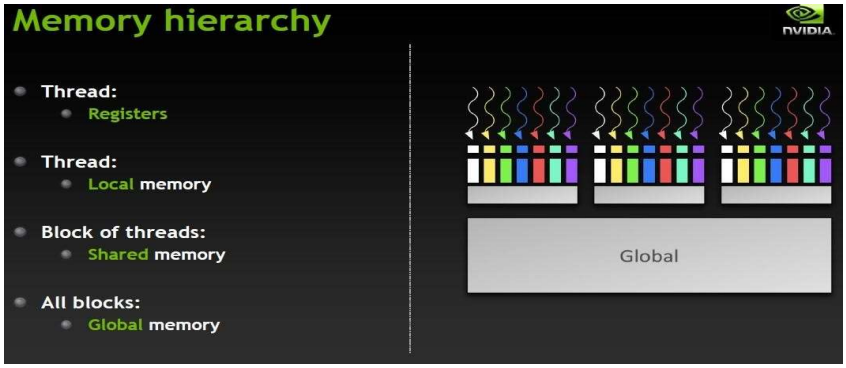
CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	What are Warps? What is the size of a Warp defined by nVidia? Ans: Group of 32 threads is called a Warp. A warp of threads can be issued for SM for execution.	K1	CO1	1.3.1 2.1.3
2	What is the bandwidth of PCI-E lanes in Core-2 series GPU architecture? Ans: 5GB/s	K1	CO1	1.3.1 2.1.3
3	What are PTX instructions? Explain the fields of the PTX instruction format. Ans: opcode.type d, a, b, c; – where d is the destination operand; a, b, and c are source operands Source operands are 32-bit or 64-bit registers or a constant value. Destinations are registers,except for store instructions	K2	CO1	1.4.1
4	List the differences between CPU and GPU	K2	CO1	1.3.1 2.2.2
5	How does thread divergence impact the performance of CUDA applications,? Ans: If some threads take the if branch and other threads take the else branch, they cannot operate in lockstep Some threads must wait for the others to execute	K2	CO1	

	Renders code at that point to be serial rather than parallel			
6	Identify the CUDAAPIcall that make sure that all previous kernel executions and memory copies have been completed. Ans: cudaDeviceSynchronize()	K3	CO2	

Part – B (3×6 = 18 Marks)

7	1) We want to use each thread to calculate two(adjacent) elements of a vector addition. Assume that variable i should be the index for the first element to be processed by a thread. Identify the correct expression for mapping the thread/block indices to data index? Justify your answer. Ans: To map thread/block indices to data indices for calculating two adjacent elements of a vector addition, you can use the following expression: **int i = blockIdx.x * blockDim.x + threadIdx.x * 2;** Here's an explanation of each part of the expression: blockIdx.x represents the index of the block in the x dimension. blockDim.x represents the number of threads per block in the x dimension. threadIdx.x represents the index of the thread in the x dimension within the block. The expression threadIdx.x * 2 is used to ensure that each thread processes two adjacent elements of the vector since it multiplies the thread index by 2. This is appropriate if each thread is responsible for computing two adjacent elements in the vector addition.	K3	CO1	1.3.1 2.1.3
8	Explain the Memory Structure of GPU processor. Ans: 	K2	CO1	1.3.1 2.1.3

9	We are processing vector a that has 64 million data. We can launch one thread for each element of a vector. If we are grouping 1024 elements in to a block and assign one block to each Streaming Multiprocessor. a)Estimate the number of threads, warps, blocks need to process the data. b)Estimate the number of Streaming Multiprocessors needed in the system. c) Estimate the storage space requirements for storing: i)single precision floating point data. ii) Double precision floating point data Ans: ANS: 64 MILLION THREADS 64 ILLION /32 = 2 MILLION WARPS Single-precision floating-point number, requiring 4 bytes of data, need around 256 million bytes, or 256 MB, of data storage space for single precision and 512 MB for Double precision data	K3	CO1	1.3.1 1.4.1 **2.2.2** **3.2.2**

Part – C (2×10 = 20 Marks)

10	Explain the with a neat block diagram the architecture of Core 2 series GPU processor.	K2	CO1	1.3.1 1.4.1 2.1.3
	(OR)			
11	Explain in detail various CUDA compute levels.	K2	CO1	1.3.1 1.4.1 2.1.3

12	Explain the process of performing vector addition using CUDA programming. Provide a step-by-step explanation of how to parallelize the addition of two large vectors on a GPU, Discuss the advantages of using CUDA for vector addition compared to a sequential CPU approach.
	Develop a CUDA program to perform addition of two vectors.
	Ans:
	_global void VecAdd(float* A, float* B, float* C, int N) {
	int i = blockDim.x * blockIdx.x + threadI dx.x;
	if (i < N)
	C[i] = A[i] + B[i];
		K3	CO2	1.3.1
				2.2.2
				2.2.3
				3.3.1

	}
	int main() {
	float* h_A = (float*)malloc(size);
	float* h_B = (float*)malloc(size);
	float* h_C = (float*)malloc(size);
	float* d_A; cudaMalloc(&d_A, size); float* d_B; cudaMalloc(&d_B, size);
	float* d_C; cudaMalloc(&d_C, size);
	// Copy vectors from host memory to
	// device memory cudaMemcpy(d_A, h_A, size,
	cudaMemcpyHostToDevice);
	cudaMemcpy(d_B, h_B, size,
	cudaMemcpyHostToDevice);
	// Invoke kernel
	int threadsPerBlock = 256;
	int blocksPerGrid = N/threadsPerBlock;
	VecAdd<<<blocksPerGrid, threadsPerBloc k>>>(d_A, d_B, d_C, N);
	// Copy result from device memory to
	// host memory cudaMemcpy(h_C, d_C, size,
	cudaMemcpyDeviceToHost);
	…
	cudaFree(d_A);
	cudaFree(d_B);
	cudaFree(d_C);			

	(OR)			
13	Describe the key steps involved in implementing Radix Sort in GPU systems, including how you would handle parallelization and memory management.Develop a CUDA program to sort list of elements in an array using Radix sort. Ans: • It has a fixed number of iterations and a consistent execution flow.	K3	CO2	1.3.1 2.2.2 2.2.3 3.3.1

- It works by sorting based on the least significant bit and then working up to the most significant bit.
- With a 32-bit integer, using a single radix bit, you will have 32 iterations of the sort, no matter how large the dataset.
- example : { 122, 10, 2, 1, 2, 22, 12, 9 }
- The binary representation of each of these would be

  122 = 01111010
  10 = 00001010
  2 = 00000010
  22 = 00010010
  12 = 00001100
  9 = 00001001

- In the first pass, all elements with a 0 in the LSB would form the first list.
- Those with a 1 as the LSB would form the second list. Thus, the two lists are

  0 = { 122, 10, 2, 22, 12 } & 1 = { 9 }
- The two lists are appended in this order, becoming

  { 122, 10, 2, 22, 12, 9 }
- The process is then repeated for bit one, generating the next two lists based on the ordering of the previous cycle:

  0 = { 12, 9 } & 1 = { 122, 10, 2, 22 }
- The combined list is then

  { 12, 9, 122, 10, 2, 22 }
- Scanning the list by bit two, we generate

  0 = { 9, 122, 10, 2, 22 } & 1 = { 12 }
  = { 9, 122, 10, 2, 22, 12 }
- The program continues until it has processed all 32 bits of the list in 32 passes.
- To build the list you need **N + 2N** memory cells.
- one for the source data, one of the **0 list**, and one of the **1 list**.

```
__device__ void radix_sort (u32 * const sort_tmp, const u32 num_lists,
const u32 num_elements, const u32 tid, u32 * const sort_tmp_0, u32 * const
sort_tmp_1)
{
// Sort into num_list, lists
// Apply radix sort on 32 bits of data
        for (u32 bit=0;bit<32;bit++)
        {
        u32 base_cnt_0 = 0;
        u32 base_cnt_1 = 0;
        for (u32 i=0; i<num_elements; i+=num_lists)
        {
        const u32 elem = sort_tmp[i+tid];
        const u32 bit_mask = (1 << bit);
if ( (elem & bit_mask) > 0 )
        {
        sort_tmp_1[base_cnt_1+tid] = elem;
        base_cnt_1+=num_lists;
        }
        else
        {
        sort_tmp_0[base_cnt_0+tid] = elem;
        base_cnt_0+=num_lists;
```

```
        }
        }
// Copy data back to source - first the zero list
for (u32 i=0; i<base_cnt_0; i+=num_lists)
        {
        sort_tmp[i+tid] = sort_tmp_0[i+tid];
        }
// Copy data back to source - then the one list
        for (u32 i=0; i<base_cnt_1; i+=num_lists)
        {
        sort_tmp[base_cnt_0+i+tid] = sort_tmp_1[i+tid];
        }
        }
        __syncthreads();
}
```

Prepared By **PAC Members** **Approved By (HOD/CSE)**

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -2** **Question Paper**						
Degree & Branch	B.E				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**18.10.2023**	**FN**
Time: 08:10 - 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	K1	What are eager evaluation and lazy evaluation?	CO2	2.1.2 2.2.2 13.1.1
2	K1	What is speculative execution?	CO2	1.4.1 2.1.2 2.2.2
3	K3	Identify the CUDA API call used to retrieve GPU device properties?.	CO2	2.1.2 13.3.2
4	K1	What is loop invariant analysis?.	CO2	2.1.2 2.2.2
5	K1	What does the term "cache coherence" refer to, and what are the available approaches to address cache coherence?	CO3	2.2.2 13.1.1 13.1.2
6	K3	A CUDA kernel performs the following operation C[z] = A[y] * B[x] Identify whether the kernel is memory bound or arithmetic bound? Justify your answer.	CO3	1.4.1 2.1.2 13.3.1

Part – B (3×6 = 18 Marks)

7	K3	Explain the role and application of CUDA ballots in GPU programming. Develop a code snippet in which CUDA ballots prove advantageous for coordinating threads. .	CO2	1.4.1 2.2.2 13.1.2 13.3.1

8	K3	Apply the concept of loop fusion for the following code snippet and demonstrate how it improves performance.
		unsigned int i,j;
		a = 0;
		for (i=0; i<100; i++)
		{
		a += b * c * i;
		}
		d = 0;
		for (j=0; j<200; j++)
		{
		d+= e * f
		}
			CO2	1.4.1
				2.2.2
				13.1.1

9	K3	Apply the concept of loop unrolling for the following loop structure demonstrate how loop unrolling will improve performance in parallel programming. for (i=1; i<=1000; i++) X[i] = X[i] + S; X is an array whose starting address is stored in the register R1 R2 contains the terminal address of an array x S is a constant stored in the register F2	CO2	1.3.1 1.4.1 **2.2.2** 13.3.1

Part – C (2×10 = 20 Marks)

10	K2	Explain the approach for error handling in CUDA programming.	CO3	1.3.1 1.4.1 2.2.2 13.3.1 13.3.2
		(OR)		
11	K2	Describe the challenges related to algorithmic aspects in CUDA programming.	CO3	1.3.1 1.4.1 2.2.2 13.3.1 13.3.2
12	K3	Create a CUDA program that employs the Binary Search technique to locate an element within an array.	CO2	1.4.1 2.1.2 2.2.2 13.3.1 13.3.2
		(OR)		
13	K3	Develop a CUDA program that populates an array with values ranging from 0 to num_elements. Additionally, set up four streams to run concurrently on four different GPU devices. The objective is to measure the following metrics for each of the four GPU devices: • The duration it takes to transfer data from the CPU to the GPU. • The time required to execute the kernel operation. • The time it takes to copy the results back from the GPU to the CPU • The total execution time of the entire operation	CO2	1.4.1 2.1.2 2.2.2 13.3.1 13.3.2

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -2** **Question Paper**						
Degree & Branch	B.E				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**.10.2023**	**FN**
Time: 08:10 - 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	K1	What are eager evaluation and lazy evaluation? Ans: In the eager evaluation model used by CPUs we stall at the first read into a1, and on each subsequent read. With the lazy evaluation model used by GPUs we stall only on consumption of the data, the additions in the third code segment, if that data is not currently available	CO2	2.1.2 2.2.2 13.1.1
2	K1	What is speculative execution? Ans: The CPU will have predicted a branch correctly, it makes sense to start executing the instruction stream at that branch address. However, this adds to the cost of branch misprediction, as now the instruction stream that has been executed has to be discarded. The optimal model for both branch prediction and speculative execution is simply to execute both paths of the branch and then commit the results when the actual branch is known.	CO2	1.4.1 2.1.2 2.2.2
3	K3	Identify the CUDA API call used to retrieve GPU device properties?.	CO2	2.1.2

		cudaError_t cudaGetDeviceProperties(struct cudaDeviceProp properties, int device);* *struct cudaDeviceProp device_0_prop;* *CUDA_CALL(cudaGetDeviceProperties(&device_0_prop, 0));*		13.3.2
4	K1	What is loop invariant analysis? Loop invariant analysis looks for expressions that are constant within the loop body and moves them outside the loop body.	CO2	2.1.2 2.2.2
5	K1	What does the term "cache coherence" refer to, and what are the available approaches to address cache coherence? Ans: Suppose two cores need to update x, because one core is assigned a debit processing task and the other a credit processing task. Both cores must have a consistent view of the memory location holding the parameter x. This is the issue of cache coherency. Write invalidate and write update protocols.	CO3	1.3.1 1.4.1 2.2.2 13.1.1 13.1.2
6	K2	A CUDA kernel performs the following operation C[z] = A[y] * B[x] Identify whether the kernel is memory bound or arithmetic bound? Justify your answer ANS: Kernel that fetches two values from memory, multiplies them, and stores the result back to memory has very low arithmetic density. C[z] = A[y] * B[x] The real work being done is the multiplication. With only one operation being performed per three memory transactions (two reads and one write), the kernel is very much memory bound	CO3	1.4.1 2.1.2 13.3.1

Part – B (3×6 = 18 Marks)

7	K3	Explain the role and application of CUDA ballots in GPU programming? Develop a code snippet in which CUDA ballots prove advantageous for coordinating threads?
		ANS:
		_ballot():
		unsigned int __ballot(int predicate);
		This function evaluates the predicate value passed to it by a given thread.
		A predicate, in this context, is simply a true or false value. If the predicate value is nonzero, it returns a value with the Nth bit set, where N is the value of the thread (threadIdx.x).
		This atomic operation can be implemented as
		C source code as follows:
		__device__ unsigned int __ballot_non_atom(int predicate)
		{
		if (predicate != 0)
		return (1 << (threadIdx.x % 32));
		else
		return 0;
		}
		The usefulness of ballot may not be immediately obvious, unless you combine it with another atomic operation, atomicOr.
		The prototype for this is
		int atomicOr(int * address, int val);
		It reads the value pointed to by address, performs a bitwise OR operation (the j operator in C) with the contents of val, and writes the value back to the address.
		It also returns the old value
		It can be used in conjunction with the __ballot function as follows:
		volatile __shared__ u32
		warp_shared_ballot[MAX_WARPS_PER_BLOCK];
		// Current warp number - divide by 32
		const u32 warp_num = threadIdx.x >> 5;
		atomicOr(&warp_shared_ballot[warp_num],
		__ballot(data[tid] > threshold));
		we use an array that can be either in shared memory or global memory, but obviously
		shared memory is preferable due to it's speed.
		We write to an array index based on the warp number, which we implicitly assume here is 32.
		Thus, each thread of every warp contributes 1 bit to the result for that warp.
		For the predicate condition, if the value in data[tid], our source data, is greater than a given threshold.
		Each thread reads one element from this dataset.
		The results of each thread are combined to form a bitwise OR of the result where thread 0 sets (or not) bit 0, thread 1 sets (or not) bit 1, etc.
			CO2	1.4.1
				2.2.2
				13.1.2
				13.3.1

8	K3	Apply the concept of loop fusion for the following code snippet and demonstrate how it improves performance.
		unsigned int i,j;
		a = 0;
		for (i=0; i<100; i++)
		{
		a += b * c * i;
			CO2	1.4.1
				2.2.2
				13.1.1

```
}
d = 0;
for (j=0; j<200; j++)
{
   d+= e * f
}
```

ANS:

loop fusion:

```
void loop_fusion_example_unfused(void)
{
unsigned int i,j;
a = 0;
for (i=0; i<100; i++) /* 100 iterations */
{
a += b * c * i;
}
d = 0;
for (j=0; j<200; j++) /* 200 iterations */
{
D+= e * f
}
}

void loop_fusion_example_fused_01(void)
{
unsigned int i; /* Notice j is eliminated */
a = 0;
d = 0;
for (i=0; i<100; i++) /* 100 iterations */
{
a +=b * c * i;
d += e * f * i;
}
for (i+100; i<200; i++) /* 100 iterations */
{
d += e * f * i;
}
}

void loop_fusion_example_fused_02(void)
{
unsigned int i; /* Notice j is eliminated */
a = 0;
d =0;
for (i=0; i<100; i++) /* 100 iterations */
   {
   a += b * c * i;
   d += e * f * i;
   d += e * f * (i*2);
   }
}
```

we have two independent calculations for results a and d.

The number of iterations required in the second calculation is more than the first.

However, the iteration space of the two calculations overlaps.

		You can, therefore, move part of the second calculation into the loop body of the first, as shown in function loop_fusion_example_fused_01. This has the effect of introducing additional, independent instructions, plus reducing the overall number of iterations, in this example, by one-third.		

9	K3	Apply the concept of loop unrolling for the following loop structure demonstrate how loop unrolling will improve performance in parallel programming.
		for (i=1; i<=1000; i++)
		X[i] = X[i] + S;
		X is an array whose starting address is stored in the register R1
		R2 contains the terminal address of an array x
		S is a constant stored in the register F2
		ANS:
		Loop: LD F0, 0(R1)
		ADDD F4, F0, F2
		SD 0(R1), F4 #1
		
		LD F6, -8(R1)
		ADDD F8, F6, F2
		SD -8(R1), F8 #2
		
		LD F10,-16(R1)
		ADDD F12,F10,F2
		SD -16(R1), F12 #3
		
		LD F14,-24(R1)
		ADDD F16,F14,F2
		SD -24(R1),F16 #4
		
		SUBI R1, R1, #32
		BENZ R1, Loop
		
		Loop: LD F0, 0(R1) 1
		stall 2
		ADDD F4, F0, F2 3
		stall 4
		stall 5
		SD 0(R1), F4 6 ;drop SUBI &BNEZ #1
		LD F6, -8(R1) 7
		stall 8
		ADDD F8, F6, F2 9
		stall 10
		stall 11
		SD -8(R1), F8 12 ;drop SUBI &BNEZ #2
		LD F10,-16(R1) 13
		stall 14
		ADDD F12,F10,F2 15
		stall 16
		stall 17
		SD -16(R1), F12 18 ;drop SUBI &BNEZ #3
		LD F14,-24(R1) 19
		stall 20
		ADDD F16,F14,F2 21
		stall 22
			CO2	1.3.1
				1.4.1
				2.2.2
				13.3.1

		stall 23 SD -24(R1),F16 24 #4 SUBI R1, R1, #32 25 BENZ R1, Loop 26 Stall 27 Loop Unrolling :Without any scheduling • It takes 27 cycles for 4 iterations 27/4 = 6.8 clock cycles per iteration • CPI =6.8/3 =2.2 Instruction cycles Loop: LD F0, 0(R1) 1 LD F6, -8(R1) 2 LD F10,-16(R1) 3 LD F14,-24(R1) 4 ADDD F4, F0, F2 5 ADDD F8, F6, F2 6 ADDD F8, F6, F2 7 ADDD F16, F14, F2 8 SD 0(R1), F4 9 SD -8(R1), F8 10 SD -16(R1), F12 11 SUBI R1, R1, #32 12 BENZ R1, Loop 13 SD 8(R1), F16 14 clock cycles per 4 iterations 14/4 = 3.5 clock cycles per iteration • CPI=3.5/3 =1.16		

Part – C (2×10 = 20 Marks)

10	K2	Explain the approach to error handling in CUDA programming.
		ANS:
		CUDA provides a set of error handling mechanisms.
		The most common macros are:
		cudaGetErrorString:
		Converts a CUDA error code into a human-readable string representation.
		cudaGetLastError:
		Returns the last error that occurred.
		cudaSuccess:
		A special value indicating that CUDA call succeeded.
		cudaError_t cudaStatus;
		cudaStatus = cudaSomeFunction();
		
		if (cudaStatus != cudaSuccess) {
		fprintf(stderr, "CUDA error: %s\n",
		
		cudaGetErrorString(cudaStatus));
		}
		Asynchronous Error Checking:
			CO3	1.3.1
				1.4.1
				2.2.2
				13.3.1
				13.3.2

		When using CUDA APIs that return error codes, you need to ensure that any previous asynchronous operations are completed before checking for errors. you can use cudaDeviceSynchronize() or other appropriate synchronization methods before checking for errors.
		// Example kernel launch
		myKernel<<<gridSize, blockSize>>>(input, output);
		
		// Synchronize the device to ensure the kernel execution is completed.
		cudaDeviceSynchronize();
		// Check for errors after synchronization.
		cudaError_t cudaStatus = cudaGetLastError();
		if (cudaStatus != cudaSuccess) {
		fprintf(stderr, "CUDA error after kernel execution: %s\n", cudaGetErrorString(cudaStatus));
		}
		Error Reporting and Debugging:
		When developing CUDA applications, it's essential to keep track of error messages.
		You can use logging libraries or simply write the error messages to standard output or error streams.
		You can make use of NVIDIA's debugging tools like NVIDIA Nsight or CUDA-GDB to analyze and debug your CUDA code effectively.
		Graceful Resource Management:
		In case of any CUDA error, it's crucial to perform proper cleanup and resource management.
		This involves releasing any allocated memory on the device, resetting the device, and freeing resources before exiting the application.
		// Release resources and clean up
		cudaFree(devicePtr);
		cudaDeviceReset();		

(OR)				
11	K2	Describe the challenges related to algorithmic aspects in CUDA programming. ANS: Thread Divergence: In CUDA, work is divided into threads, and threads are grouped into blocks. When threads within a block follow different execution paths (e.g., if-else statements), it can lead to thread divergence. Thread divergence negatively impacts performance because the GPU needs to execute all the branches taken by different threads, serializing the process Memory Access Patterns: Memory access patterns play a crucial role in GPU performance. Irregular memory accesses, such as non-coalesced reads/writes or random memory accesses, can result in lower memory throughput and increased memory latency. Optimizing memory access patterns can significantly improve the overall performance Workload Imbalance: Load balancing is essential to keep all GPU cores busy. If some threads within a block finish their work earlier than others, they will be idle until all threads have completed their tasks. Identifying and minimizing workload imbalances can lead to better GPU utilization	CO3	1.3.1 1.4.1 2.2.2 13.3.1 13.3.2

		Overhead of Kernel Launches: Kernel launches on CUDA GPUs come with some overhead. It is essential to design algorithms that minimize the number of kernel launches and the data transfers between the host (CPU) and the device (GPU) to achieve better performance. Synchronization: Synchronization points, such as barriers or locks, can introduce performance bottlenecks in GPU computation. Efficiently managing synchronization in CUDA programs is essential to avoid unnecessary stalls and maximize parallelism. Data Dependencies: Dependencies between data elements can hinder parallelism and stall GPU execution. Identifying and minimizing data dependencies through techniques like loop unrolling and loop tiling can improve GPU performance Thread/Block Granularity: Choosing the appropriate thread and block granularity is critical for achieving optimal performance. If the granularity is too fine, the overhead of managing threads and blocks can outweigh the computational benefits. On the other hand, if it's too coarse, some GPU resources may be underutilized. Memory Allocation and Deallocation: Frequent memory allocation and deallocation on the GPU can impact performance. Reusing memory buffers whenever possible can reduce the overhead associated with memory management. Precision of Computations: GPUs often support different precisions (e.g., single-precision and half-precision floating-point arithmetic). Choosing the right precision for computations can impact both performance and numerical accuracy. Scalability: While GPUs offer massive parallelism, not all algorithms are inherently parallelizable. Ensuring scalability of algorithms to fully utilize the available GPU resources is crucial for achieving optimal performance.		
12	K3	Create a CUDA program that employs the Binary Search technique to locate an element within an array. ANS: The search we have two options: a binary search.	CO2	1.4.1 2.1.2 2.2.2 13.3.1 13.3.2

A binary search takes advantage of the fact we have a sorted list of samples from the previous step.

It works by dividing the list into two halves and asking whether the value it seeks is in the top or bottom half of the dataset.

It then divides the list again and again until such time as it finds the value.

The worst case sort time for a binary search is log2(N).

```
#include <iostream>

#include <cstdio>


// Kernel function for binary search

__global__ void binarySearchKernel(int *arr, int target, int left, int right, int *result) {

   int tid = blockIdx.x * blockDim.x + threadIdx.x;

while (left <= right) {

      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {

         atomicExch(result, mid); // Store the result in a thread-safe manner

         return;

      } else if (arr[mid] < target) {

         left = mid + 1;

      } else {

         right = mid - 1;

      }

   }

}

int main() {

   const int arraySize = 1024;

   const int target = 42;
```

```
    int *hostArray, *deviceArray, *deviceResult;


    // Allocate memory on host and device
    hostArray = new int[arraySize];
    cudaMalloc(&deviceArray, arraySize * sizeof(int));
    cudaMalloc(&deviceResult, sizeof(int));


// Initialize the sorted array on the host
    for (int i = 0; i < arraySize; ++i) {
        hostArray[i] = i * 2;
    }
// Copy data from host to device
    cudaMemcpy(deviceArray, hostArray, arraySize * sizeof(int),
cudaMemcpyHostToDevice);


    // Set up grid and block dimensions
    int threadsPerBlock = 256;
    int blocksPerGrid = (arraySize + threadsPerBlock - 1) / threadsPerBlock;
// Launch kernel
    binarySearchKernel<<<blocksPerGrid, threadsPerBlock>>>(deviceArray,
target, 0, arraySize - 1, deviceResult);
// Copy result from device to host
    int result;
    cudaMemcpy(&result, deviceResult, sizeof(int),
cudaMemcpyDeviceToHost);
if (result != -1) {
        std::cout << "Element " << target << " found at index " << result <<
std::endl;
    } else {
        std::cout << "Element " << target << " not found in the array." <<
std::endl;
```

		}
		// Clean up
		delete[] hostArray;
		cudaFree(deviceArray);
		cudaFree(deviceResult);
		return 0;
		}		

(OR)				

13	K3	Develop a CUDA program that populates an array with values ranging from 0 to num_elements. Additionally, set up four streams to run concurrently on four different GPU devices? The objective is to measure the following metrics for each of the four GPU devices:
		• The duration it takes to transfer data from the CPU to the GPU.
		• The time required to execute the kernel operation.
		• The time it takes to copy the results back from the GPU to the CPU
		• The total execution time of the entire operation
		ANS:
		Streams are virtual work queues on the GPU.
		They are used for asynchronous operation i.e, when you would like the GPU to operate separately from the CPU.
		By creating a stream you can push work and events into the stream which will then execute the work in the order in which it is pushed into the stream.
		Streams and events are associated with the GPU context in which they were created.
		void fill_array(u32 * data, const u32 num_elements)
		{
		for (u32 i=0; i< num_elements; i++)
		{
		data[i] = i;
		}
		}
		void check_array(char * device_prefix, u32 * data, const u32 num_elements)
		{
		bool error_found = false;
		for (u32 i=0; i< num_elements; i++)
		{
		if (data[i] !=(i*2))
		{
		printf("%sError: %u %u", device_prefix, i, data[i]);
		error_found = true;
		}
		}
		if (error_found ==false)
		printf("%sArray check passed", device_prefix);
		}
		// Define maximum number of supported devices
		#define MAX_NUM_DEVICES (4)
		// Define the number of elements to use in the array
			CO2	1.4.1
				2.1.2
				2.2.2
				13.3.1
				13.3.2

```
        #define NUM_ELEM (1024*1024*8)
// Define one stream per GPU
        cudaStream_t stream[MAX_NUM_DEVICES];
// Define a string to prefix output messages with so
// we know which GPU generated it
        char device_prefix[MAX_NUM_DEVICES][300];
// Define one working array per device, on the device
        u32 * gpu_data[MAX_NUM_DEVICES];
// Define CPU source and destination arrays, one per GPU
        u32 * cpu_src_data[MAX_NUM_DEVICES];
        u32 * cpu_dest_data[MAX_NUM_DEVICES]
// Generate a prefix for all screen messages
        struct cudaDeviceProp device_prop;
        CUDA_CALL(cudaGetDeviceProperties(&device_prop,
device_num));
        sprintf(&device_prefix[device_num][0], "\nID:%d %s:", device_num,
                                                device_prop.name);

// Create a new stream on that device
        CUDA_CALL(cudaStreamCreate(&stream[device_num]));

// Allocate memory on the GPU
        CUDA_CALL(cudaMalloc((void**)&gpu_data[device_num],
                                        single_gpu_chunk_size));
// Allocate page locked memory on the CPU
         CUDA_CALL(cudaMallocHost((void                                **)
&cpu_src_data[device_num],
                                                single_gpu_chunk_size));
        CUDA_CALL(cudaMallocHost((void
**)&cpu_dest_data[device_num],
                                        single_gpu_chunk_size));

// Fill it with a known pattern
fill_array(cpu_src_data[device_num], NUM_ELEM);

// Copy a chunk of data from the CPU to the GPU  asynchronous
        CUDA_CALL(cudaMemcpyAsync(gpu_data[device_num],
                cpu_src_data[device_num], single_gpu_chunk_size,
                cudaMemcpyHostToDevice, stream[device_num]));
// Invoke the GPU kernel using the newly created  stream - asynchronous
invocation
                gpu_test_kernel<<<num_blocks,
                num_threads,
                shared_memory_usage,
                stream[device_num]>>>(gpu_data[device_num]);

        cuda_error_check(device_prefix[device_num],
                "Failed to invoke gpu_test_kernel");
// Now push memory copies to the host into the streams
// Copy a chunk of data from the GPU to the CPU asynchronous
        CUDA_CALL(cudaMemcpyAsync(cpu_dest_data[device_num],
                        gpu_data[device_num],
                        single_gpu_chunk_size,
                        cudaMemcpyDeviceToHost,
                        stream[device_num]));
```

```
}
// Process the data as it comes back from the GPUs  Overlaps CPU execution
with GPU execution
        for (int device_num=0;device_num < num_devices;device_num++)
{
// Select the correct device
CUDA_CALL(cudaSetDevice(device_num));
//Wait for all commands in the stream to complete
CUDA_CALL(cudaStreamSynchronize(stream[device_num]));
// GPU data and stream are now used, so  clear them up
        CUDA_CALL(cudaStreamDestroy(stream[device_num]));
        CUDA_CALL(cudaFree(gpu_data[device_num]));

// Data has now arrived in  cpu_dest_data[device_num]
        check_array(                                device_prefix[device_num],
cpu_dest_data[device_num], NUM_ELEM);

// Clean up CPU allocations
        CUDA_CALL(cudaFreeHost(cpu_src_data[device_num]));
        CUDA_CALL(cudaFreeHost(cpu_dest_data[device_num]));

// Release the device context
        CUDA_CALL(cudaDeviceReset());
}
}
```

it checks the contents and then frees the GPU and CPU resources associated with each stream.

we need to add some timing code to see how long each kernel takes in practice.

Add events to the work queue.

Now events are special in that we can query an event regardless of the currently selected GPU

we need to declare a start and stop event:

```
// Define a start and stop event per stream
        cudaEvent_t kernel_start_event[MAX_NUM_DEVICES];
        cudaEvent_t memcpy_to_start_event[MAX_NUM_DEVICES];
        cudaEvent_t memcpy_from_start_event[MAX_NUM_DEVICES];
        cudaEvent_t memcpy_from_stop_event[MAX_NUM_DEVICES];
```

Finally, we need to get the elapsed time and print it to the screen:

```
// Wait for all commands in the stream to complete
        CUDA_CALL(cudaStreamSynchronize(stream[device_num]));

// Get the elapsed time between the copy  and kernel start
        CUDA_CALL(cudaEventElapsedTime(&time_copy_to_ms,memcpy
_to      _start_event[device_num], kernel_start_event[device_num]));

// Get the elapsed time between the kernel start and copy back start
        CUDA_CALL(cudaEventElapsedTime(&time_kernel_ms,
        kernel_start_event[device_num],memcpy_from_start_event[device
        _num]));
Get the elapsed time between the copy back start  and copy back start
        CUDA_CALL(cudaEventElapsedTime(&time_copy_from_ms,memc
py_from_       start_event[de vice_num],
memcpy_from_stop_event[device_num]));

// Get the elapsed time between the overall start and stop events
```

```
        CUDA_CALL(cudaEventElapsedTime(&time_exec_ms,
        memcpy_to_start_event[device_num],
        memcpy_from_stop_event[device_num]));

// Print the elapsed time
        const float gpu_time = (time_copy_to_ms þ time_kernel_ms +
        time_copy_from_ms);
        printf("%sCopy To : %.2f ms",
        device_prefix[device_num], time_copy_to_ms);
        printf("%sKernel : %.2f ms",
        device_prefix[device_num], time_kernel_ms);
printf("%sCopy Back : %.2f ms",  device_prefix[device_num],
                time_copy_from_ms);
        printf("%sComponent Time : %.2f ms", device_prefix[device_num],
                        gpu_time);
        printf("%sExecution Time : %.2f ms", device_prefix[device_num],
        time_exec_ms);
        printf("\n");
When we run the program we see the following result:
ID:0 GeForce GTX 470:Copy To : 20.22 ms
ID:0 GeForce GTX 470:Kernel : 4883.55 ms
ID:0 GeForce GTX 470:Copy Back : 10.01 ms
ID:0 GeForce GTX 470:Component Time : 4913.78 ms
ID:0 GeForce GTX 470:Execution Time : 4913.78 ms
ID:0 GeForce GTX 470:Array check passed

ID:1 GeForce 9800 GT:Copy To : 20.77 ms
ID:1 GeForce 9800 GT:Kernel : 25279.57 ms
ID:1 GeForce 9800 GT:Copy Back : 10.02 ms
ID:1 GeForce 9800 GT:Component Time : 25310.37 ms
ID:1 GeForce 9800 GT:Execution Time : 25310.37 ms
ID:1 GeForce 9800 GT:Array check passed
When we run the program we see the following result:
ID:0 GeForce GTX 470:Copy To : 20.22 ms
ID:0 GeForce GTX 470:Kernel : 4883.55 ms
ID:0 GeForce GTX 470:Copy Back : 10.01 ms
ID:0 GeForce GTX 470:Component Time : 4913.78 ms
ID:0 GeForce GTX 470:Execution Time : 4913.78 ms
ID:0 GeForce GTX 470:Array check passed

ID:1 GeForce 9800 GT:Copy To : 20.77 ms
ID:1 GeForce 9800 GT:Kernel : 25279.57 ms
ID:1 GeForce 9800 GT:Copy Back : 10.02 ms
ID:1 GeForce 9800 GT:Component Time : 25310.37 ms
ID:1 GeForce 9800 GT:Execution Time : 25310.37 ms
ID:1 GeForce 9800 GT:Array check passed
```

Prepared By **PAC Members** **Approved By**

(HOD/CSE)

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -3** **Question Paper**						
Degree & Branch	B.E				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**10.11.2023**	**FN**
Time: 08:10 - 09:40 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	K1	What are OpenCL Contexts? ANS: a context is an abstract environment within which coordination and memory management for kernel execution is well defined a context must be configured that enables the host to pass commands and data to the device. A context coordinates : • host-device interaction, • manages the memory objects available to the devices, • keeps track of the programs and kernels that are created for each device.	CO4	2.1.2 2.2.2 13.1.1
2	K1	What is the significance of OpenCL API call get_global_id(0)? ANS: The call to get_global_id(0) allows the programmer to make use of the position of the current work-item to access a unique element in the array. The parameter "0" to the get_global_id() function assumes that we have specified a one-dimensional configuration of work-items, and therefore only need its ID in the first dimension.	CO4	1.4.1 2.1.2 2.2.2
3	K3	Identify the OpenCL API call used for finding the available platforms on the system and provide the corresponding syntax for this operation?	CO4	2.1.2 13.3.2

		ANS: The OpenCL API function The API call clGetPlatformIDs() is used to discover the set of available OpenCL platforms for a given system. clGetPlatformIDs (cl_uint num_entries, cl_platform_id *platforms, cl_uint *num_platforms)		
4	K1	What do memory objects refer to, and what are the various types of memory objects found in OpenCL? ANS: In order for data to be transferred to a device, it must first be encapsulated as a memory object. In order for output data to be generated, space must also be allocated and encapsulated as a memory object. OpenCL defines three types of memory objects: buffers, images, and pipes	CO4	2.1.2 2.2.2
5	K1	What is the difference between the exclusive and inclusive variants of prefixsum algorithm? ANS: There are two main variants of the Prefix Sum algorithm: exclusive and inclusive • In an exclusive Prefix Sum, the element at index i in the output array is the sum of all elements in the input array up to, but not including, index i. • In an inclusive Prefix Sum, the element at index i in the output array is the sum of all elements in the input array up to and including index i.	CO4	2.2.2 13.1.1 13.1.2
6	K3	Identify the OpenCL API call for creating a command-queue and include the relevant syntax for this action? ANS: The API call clCreateCommandQueueWithProperties() is used to create a command-queue and associate it with a device. cl_command_queue clCreateCommandQueueWithProperties (cl_context context, cl_device_id device, cl_command_queue_properties properties, cl_int* errcode_ret)	CO4	1.4.1 2.1.2 13.3.1

Part – B (3×6 = 18 Marks)

7	K2	. Explain the key steps involved in running a basic OpenCL application. ANS: The main steps to execute a simple OpenCL application are as follows: 1.Discovering the platform and devices 2. Creating a context 3. Creating a command-queue per device 4. Creating memory objects (buffers) to hold data 5. Copying the input data onto the device 6. Creating and compiling a program from the OpenCL C source code 7. Extracting the kernel from the program 8. Executing the kernel 9. Copying output data back to the host 10. Releasing the OpenCL resource	CO4	1.4.1 2.2.2 13.1.2 13.3.1
8	K3	Explain the key features of OpenCL? ANS: Key features of OpenCL : Platform independence: OpenCL is designed to be cross-platform, allowing developers to write code that can run on a wide range of devices without modification. Heterogeneous computing: OpenCL enables developers to harness the power of different types of hardware, such as CPUs and GPUs, to perform computations in parallel. Parallel programming model: OpenCL provides a programming model based on data parallelism, where tasks are divided into smaller work items that can be executed simultaneously on different compute units. Host-device interaction: OpenCL allows seamless interaction between the host (usually the CPU) and the devices (such as GPUs or other accelerators) through its API. Memory management: OpenCL provides mechanisms for managing memory across different devices and moving data efficiently between the host and the devices. Scalability: OpenCL is designed to scale from mobile and embedded devices to high-performance computing clusters.	CO4	1.4.1 2.2.2 13.1.1
9	K3	What are the key steps involved in image rotation using OpenCL? ANS: Rotation is a common image processing routine with applications in matching, alignment, and other image-based algorithms. The input to an image rotation routine is an image, the rotation angle θ, and	CO4	1.3.1 1.4.1 **2.2.2** 13.3.1

		a point about which rotation is done. The coordinates of a point (x, y) when rotated by an angle θ around (x0 , y0) become (x′, y′), as shown by the following equations. $x' = \cos\theta(x - x0) + \sin\theta(y - y0)$ $y' = -\sin\theta(x - x0) + \cos\theta(y - y0)$ From the equations, it is clear that the pixel that will be stored in each output location (x′, y′) can be computed independently Each work-item correspond to an output location, we can intuitively map each work-item's global ID to (x′, y′) in the previous equations. We can also determine (x0 , y0), which corresponds to the center of the image, as soon as we load the image from disk. We have two equations and two unknowns, which allows us to compute the location read by each work-item when computing the rotation: $x = x' \cos\theta - y' \sin\theta + x0$, $y = x' \sin\theta + y' \cos\theta + y0$. This corresponds to the following OpenCL C pseudocode: gidx = get_global_id(0) gidy = get_global_id(1) x0 = width/2 y0 = height/2 x = gidx*cos(theta) - gidy*sin(theta) + x0 y = gidx*sin(theta) + gidy*cos(theta) + y0		

Part – C (2×10 = 20 Marks)

10	K3	Create a vector addition program using OpenCL to efficiently compute the sum of two arrays on a GPU?
		ANS:
		`// This program implements a vector addition using OpenCL`
		`// System includes`
		`#include <stdio.h>`
		`#include <stdlib.h>`
		`// OpenCL includes`
		`#include <CL/cl.h>`
		`// OpenCL kernel to perform an element-wise addition`
		`const char* programSource =`
		`"__kernel \n"`
		`"void vecadd(__global int *A, \n"`
		`" __global int *B, \n"`
		`" __global int *C) \n"`
			CO4	1.3.1
				1.4.1
				2.2.2
				13.3.1
				13.3.2

```
15  "{                                                          \n"
16  "                                                           \n"
17  "  // Get the work-item's unique ID                         \n"
18  "  int idx = get_global_id(0);                              \n"
19  "                                                           \n"
20  "  // Add the corresponding locations of                    \n"
21  "  // 'A' and 'B', and store the result in 'C'.             \n"
22  "  C[idx] = A[idx] + B[idx];                                \n"
23  "}                                                          \n"
24  ;
25
26  int main() {
27      // This code executes on the OpenCL host
28
29      // Elements in each array
30      const int elements = 2048;
31
32      // Compute the size of the data
33      size_t datasize = sizeof(int)*elements;
34
35      // Allocate space for input/output host data
36      int *A = (int*)malloc(datasize); // Input array
37      int *B = (int*)malloc(datasize); // Input array
38      int *C = (int*)malloc(datasize); // Output array
39
40      // Initialize the input data

41      int i;
42      for(i = 0; i < elements; i++) {
43          A[i] = i;
44          B[i] = i;
45      }
46
47      // Use this to check the output of each API call
48      cl_int status;
49
50      // Get the first platform
51      cl_platform_id platform;
52      status = clGetPlatformIDs(1, &platform, NULL);
53
54      // Get the first device
55      cl_device_id device;
56      status = clGetDeviceIDs(platform, CL_DEVICE_TYPE_ALL, 1, &device,
            NULL);
57
58      // Create a context and associate it with the device
59      cl_context context = clCreateContext(NULL, 1, &device, NULL, NULL,
            &status);
60
61      // Create a command-queue and associate it with the device
62      cl_command_queue cmdQueue = clCreateCommandQueueWithProperties
            (context, device, 0, &status);
63
64      // Allocate two input buffers and one output buffer for the three
            vectors in the vector addition
65      cl_mem bufA = clCreateBuffer(context, CL_MEM_READ_ONLY, datasize,
            NULL, &status);
66      cl_mem bufB = clCreateBuffer(context, CL_MEM_READ_ONLY, datasize,
            NULL, &status);
67      cl_mem bufC = clCreateBuffer(context, CL_MEM_WRITE_ONLY,
            datasize, NULL, &status);
68
69      // // Write data from the input arrays to the buffers
70      status = clEnqueueWriteBuffer(cmdQueue, bufA, CL_FALSE, 0,
            datasize, A, 0, NULL, NULL);
71      status = clEnqueueWriteBuffer(cmdQueue, bufB, CL_FALSE, 0,
            datasize, B, 0, NULL, NULL);
72
73      // Create a program with source code
74      cl_program program = clCreateProgramWithSource(context, 1,
            (const char**)&programSource, NULL, &status);
75
76      // Build (compile) the program for the device
77      status = clBuildProgram(program, 1, &device, NULL, NULL, NULL);
78
79      // Create the vector addition kernel
80      cl_kernel kernel = clCreateKernel(program, "vecadd", &status);
81
82      // Set the kernel arguments
83      status = clSetKernelArg(kernel, 0, sizeof(cl_mem), &bufA);
84      status = clSetKernelArg(kernel, 1, sizeof(cl_mem), &bufB);
85      status = clSetKernelArg(kernel, 2, sizeof(cl_mem), &bufC);
```

```
86
87      // Define an index space of work-items for execution.
88      // A work-group size is not required, but can be used.
89      size_t indexSpaceSize[1], workGroupSize[1];
90
91      // There are 'elements' work-items
92      indexSpaceSize[0] = elements;
93      workGroupSize[0] = 256;
94
95      // Execute the kernel
96      status = clEnqueueNDRangeKernel(cmdQueue, kernel, 1, NULL,
            indexSpaceSize, workGroupSize, 0, NULL, NULL);
97
98      // Read the device output buffer to the host output array
99      status = clEnqueueReadBuffer(cmdQueue, bufC, CL_TRUE, 0,
            datasize, C, 0, NULL, NULL);
100
101     // Free OpenCL resources
102     clReleaseKernel(kernel);
103     clReleaseProgram(program);
104     clReleaseCommandQueue(cmdQueue);
105     clReleaseMemObject(bufA);
106     clReleaseMemObject(bufB);
```

(OR)

11	K3	Implement a parallel histogram calculation using OpenCL, and what are some key considerations and challenges when optimizing the performance of your OpenCL histogram kernel for different GPU architectures?
		ANS:
		A histogram is used to count or visualize the frequency of data (i.e. the number of occurrences) over units of discrete intervals, called bins. Histograms have many applications within data and image processing.
		In this example, we will create a histogram of the frequency of pixel values within a 256-bit image
		
		A histogram is used to count or visualize the frequency of data (i.e. the number of occurrences) over units of discrete intervals, called bins. Histograms have many applications within data and image processing.
		In this example, we will create a histogram of the frequency of pixel values within a 256-bit image.
		
		Conceptually, the histogram algorithm itself is very simple. In the case where each value corresponds to a bin, a histogram could be computed as follows:
		
		int histogram[HIST_BINS]
		main( ) {
		for (each input value) {
		histogram[value]++
		}
		}
		
		The pseudocode below, which could be used to run a multithreaded version of a histogram computation.
		int histogram[HIST_BINS]
		createHistogram( ) {
		int localHistogram[HIST_BINS]
		for (each of my values) {
		localHistogram[value]++
		}
		for (each bin) {
		atomic_add(histogram[bin], localHistogram[bin])
		}
		}
		main( ) {
			CO4	1.3.1
				1.4.1
				2.2.2
				13.3.1
				13.3.2

for (number of threads) {
spawn_thread(createHistogram)
}
}
The implementation of the histogram algorithm has five steps:
1. Initialize the local histogram bins to zero (Line 14).
2. Synchronize to ensure that all updates have completed (Line 23).
3. Compute the local histogram (Line 26).
4. Synchronize again to ensure that all updates have completed (Line 35).
5. Write the local histogram out to global memory (Line 39).

```
1
2  #define HIST_BINS 256
3
4  __kernel
5  void histogram(__global int *data,
6                 int numData,
7                 __global int *histogram)
8  {
9     __local int localHistogram[HIST_BINS];
10    int lid = get_local_id(0);
11    int gid = get_global_id(0);
12
13    /* Initialize local histogram to zero */
14    for (int i = lid;
15         i < HIST_BINS;
16         i += get_local_size(0))
17    {
18       localHistogram[i] = 0;
19    }
20
21    /* Wait until all work-items within
22     * the work-group have completed their stores */
23    barrier(CLK_LOCAL_MEM_FENCE);
24
25    /* Compute local histogram */
26    for (int i = gid;
27         i < numData;
28         i += get_global_size(0))
29    {
30       atomic_add(&localHistogram[data[i]], 1);
31    }
32
33    /* Wait until all work-items within
34     * the work-group have completed their stores */
35    barrier(CLK_LOCAL_MEM_FENCE);
36
```

```
37      /* Write the local histogram out to
38       * the global histogram */
39      for (int i = lid;
40           i < HIST_BINS;
41           i += get_local_size(0))
42      {
43         atomic_add(&histogram[i], localHistogram[i]);
44      }
45  }
```

- In step 1, we stride by the work-group size to initialize each value in the local histogram to zero.
- This allows code to change work-group sizes, potentially as a performance optimization, and still remain functionally correct.
- Step 3 uses the same technique to read from global memory.
- Steps 2 and 4 use a barrier to synchronize between steps, and specify a memory fence to ensure that all work-items in the work-group have the same view of memory before proceeding.
- step 5 again uses the technique to write out of local memory

12	K3	Develop an OpenCLprogram to perform matrix multiplication of two square matrices, A and B, both of size N x N. Ans: Matrix multiplication is a computationally intensive task that can be effectively parallelized using OpenCL. The algorithm involves breaking down the matrices into smaller work units and distributing the computation across available processing units (e.g., GPU cores) to achieve parallelism. The standard matrix multiplication for matrices A (size MxK) and B (size KxN) to produce matrix C (size MxN) can be described as follows: for i = 0 to M for j = 0 to N C[i][j] = 0 for k = 0 to K C[i][j] += A[i][k] * B[k][j]. Include necessary headers and define the matrix dimensions (M, K, N) and the corresponding memory sizes. Define the matrices A, B, and C, and allocate memory for them on the host (CPU). Create an OpenCL context and command queue to interact with the device (e.g., GPU). Create OpenCL memory objects (buffers) to hold matrices A, B, and C on the device. copy matrices A and B from host memory to device memory.	CO5	1.4.1 2.1.2 2.2.2 13.3.1 13.3.2

Load and build the OpenCL kernel from a string representation.
Set kernel arguments to pass the matrices A, B, and C to the kernel function.
Define global and local work sizes to specify how the computation is distributed among work items (threads).
Enqueue the kernel for execution on the device.
Read the result matrix C from the device back to the host.
Clean up OpenCL resources.

```
// Include necessary headers and define the matrix dimensions (M, K, N) and the corresponding memory sizes
#include <CL/cl.h>
#define M 1024
#define K 1024
#define N 1024
//Define the matrices A, B, and C, and allocate memory for them on the host (CPU)
float A[M * K], B[K * N], C[M * N];
//Create an OpenCL context and command queue to interact with the device (e.g., GPU).
cl_context context = clCreateContext(NULL, 1, &device_id, NULL, NULL, &err);
cl_command_queue queue = clCreateCommandQueue(context, device_id, 0, &err);

//Create OpenCL memory objects (buffers) to hold matrices A, B, and C on the device.
cl_mem buffer_A = clCreateBuffer(context, CL_MEM_READ_ONLY, sizeof(float) * M
    * K, NULL, &err);
cl_mem buffer_B = clCreateBuffer(context, CL_MEM_READ_ONLY, sizeof(float) * K * N,
NULL, &err);
cl_mem buffer_C = clCreateBuffer(context, CL_MEM_WRITE_ONLY, sizeof(float) *
            M * N, NULL, &err);
//copy matrices A and B from host memory to device memory.
clEnqueueWriteBuffer(queue, buffer_A, CL_TRUE, 0, sizeof(float) * M * K, A, 0, NULL, NULL);
clEnqueueWriteBuffer(queue, buffer_B, CL_TRUE, 0, sizeof(float) * K * N, B, 0, NULL, NULL);
//Load and build the OpenCL kernel from a string representation.
```

```
const char* kernel_source =
  "__kernel void matrixMultiplication(__global float* A, __global
  float* B, __global float* C) {"
  "  int i = get_global_id(0);"
  "  int j = get_global_id(1);"
  "  float sum = 0.0;"
  "  for (int k = 0; k < K; ++k) {"
  "    sum += A[i * K + k] * B[k * N + j];"
  "  }"
  "  C[i * N + j] = sum;"
  "}";
cl_program program = clCreateProgramWithSource(context, 1,
  &kernel_source, NULL, &err);
clBuildProgram(program, 1, &device_id, NULL, NULL, NULL);
cl_kernel kernel = clCreateKernel(program, "matrixMultiplication",
  &err);
//Set kernel arguments to pass the matrices A, B, and C to the kernel
  function.
clSetKernelArg(kernel, 0, sizeof(cl_mem), &buffer_A);
clSetKernelArg(kernel, 1, sizeof(cl_mem), &buffer_B);
clSetKernelArg(kernel, 2, sizeof(cl_mem), &buffer_C);
//Define global and local work sizes to specify how the computation is
  distributed among work items (threads)
size_t global_size[2] = {M, N};
size_t local_size[2] = {16, 16}; // For example, use a 16x16 workgroup
  size
//Enqueue the kernel for execution on the device.
clEnqueueNDRangeKernel(queue, kernel, 2, NULL, global_size,
                                                local_size, 0, NULL, NULL);
//Read the result matrix C from the device back to the host.
clEnqueueReadBuffer(queue, buffer_C, CL_TRUE, 0, sizeof(float) * M
  *                              N, C, 0, NULL, NULL);
//Clean up OpenCL resources.
clReleaseMemObject(buffer_A);
clReleaseMemObject(buffer_B);
clReleaseMemObject(buffer_C);
clReleaseProgram(program);
clReleaseKernel(kernel);
clReleaseCommandQueue(queue);
clReleaseContext(context);
```

		(OR)		
13	K3	Develop an OpenCL program for performing Prefix Sum (also known as Scan) on an array using parallel computing techniques? Ans: • The Prefix Sum algorithm is used to calculate the cumulative sum of elements in an input array. • Given an input array A of n elements, the Prefix Sum algorithm produces an output array B, where each element B[i] is the sum of all elements in A up to and including A[i]. The Prefix Sum algorithm is used to calculate the cumulative sum of elements in an input array. Given an input array A of n elements, the Prefix Sum algorithm produces an output array B, where each element B[i] is the sum of all elements in A up to and including A[i]. There are two main variants of the Prefix Sum algorithm: exclusive and inclusive In an exclusive Prefix Sum, the element at index i in the output array is the sum of all elements in the input array up to, but not including, index i. In an inclusive Prefix Sum, the element at index i in the output array is the sum of all elements in the input array up to and including index i. Consider the exclusive Prefix Sum algorithm. The algorithm can be efficiently parallelized using OpenCL to take advantage of the massive parallelism offered by GPUs. Algorithm: Exclusive Prefix Sum using OpenCL Input: A - Input array of size n. Output: B - Output array of size n. Create two OpenCL buffers to hold the input and output arrays A and B. Load the OpenCL kernel code to perform the Prefix Sum calculation. Set up the OpenCL context, device, and command queue. Enqueue the input data from the host (CPU) to the device (GPU). Set the kernel arguments (input and output buffers) for the Prefix Sum kernel. Launch the OpenCL kernel to perform the Prefix Sum calculation in parallel. Enqueue the result (output array B) from the device to the host. Release OpenCL resources. #include <CL/cl.hpp> #include <vector>	CO5	1.4.1 2.1.2 2.2.2 13.3.1 13.3.2

```
#define N 256 // Total number of elements (should be a power of 2)
#define WORKGROUP_SIZE 64 // Workgroup size (usually a power
   of 2)

int main() {
   // Initialize input data (example)
   std::vector<int> A(N);
   // Populate A with data...

 // Create OpenCL context, device, and command queue (error checking
   omitted for simplicity)
   cl::Context context;
   cl::Device device;
   cl::CommandQueue queue;

// Create buffers for input and output data
   cl::Buffer bufferA(context, CL_MEM_READ_ONLY, N *
   sizeof(int));
   cl::Buffer bufferB(context, CL_MEM_WRITE_ONLY, N *
   sizeof(int));

/ Load the OpenCL kernel code from file (PrefixSum.cl)

   // Read the kernel source code from a file
   std::ifstream kernelFile("PrefixSum.cl");
   std::string kernelSource(std::istreambuf_iterator<char>(kernelFile),
   (std::istreambuf_iterator<char>()));
   cl::Program::Sources sources(1,
   std::make_pair(kernelSource.c_str(), kernelSource.length()));
   // Create the program from the source code
   cl::Program program(context, sources);

   // Build the program for the selected device
   program.build({ device });

// Create the kernel
   cl::Kernel kernel(program, "prefixSum");

   // Enqueue input data to the device
   queue.enqueueWriteBuffer(bufferA, CL_TRUE, 0, N * sizeof(int),
   A.data());
```

```
    // Set kernel arguments
    kernel.setArg(0, bufferA);
    kernel.setArg(1, bufferB);
    kernel.setArg(2, N);

    // Define the global and local work sizes
    cl::NDRange globalSize(N);
    cl::NDRange localSize(WORKGROUP_SIZE);

// Launch the kernel
    queue.enqueueNDRangeKernel(kernel, cl::NullRange, globalSize,
    localSize);

    // Enqueue the result from device to host
    std::vector<int> B(N);
    queue.enqueueReadBuffer(bufferB, CL_TRUE, 0, N * sizeof(int),
    B.data());

    // Output the result
    for (int i = 0; i < N; i++) {
        std::cout << "B[" << i << "] = " << B[i] << std::endl;
    }
    return 0;
}
__kernel void prefixSum(__global const int* A, __global int* B, const
    int n) {
    // Local memory for each workgroup
    __local int temp[N]; // N should be a power of 2, usually set to the
    workgroup size
    int gid = get_global_id(0);
    int lid = get_local_id(0);
    int lsize = get_local_size(0);

    // Load data from global memory to local memory
    temp[lid] = A[gid];
// Perform reduction in local memory
    for (int stride = 1; stride < lsize; stride *= 2) {
        barrier(CLK_LOCAL_MEM_FENCE);
        if (lid >= stride) {
            temp[lid] += temp[lid - stride];
        }
```

```
    }
    // Perform post-reduction in local memory
    barrier(CLK_LOCAL_MEM_FENCE);

    // Write the result to global memory
    B[gid] = (gid == 0) ? 0 : temp[lid - 1];
}
```

Register Number									

Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam – 603 110 (An Autonomous Institution, Affiliated to Anna University, Chennai)						
Computer Science and Engineering **Continuous Assessment Test -1** **Question Paper**						
Degree & Branch	B.E CSE				**Semester**	VII
Subject Code & Name	UCS1727– GPU Computing				**Regulation:**	**2018**
Academic Year	2023-2024 ODD	**Batch**	2020-2024	**Date**	**10.11.2023**	**FN**
Time: 08:15 - 09:45 AM (90 Minutes)	**Answer All Questions**				**Maximum: 50 Marks**	

(K1: Remembering, K2: Understanding, K3: Applying, K4: Analyzing, K5: Evaluating)

CO1:	Understand GPU architecture (K2)
CO2:	Write programs using CUDA, identify issues and debug them (K3)
CO3:	Implement efficient algorithms in GPUs for common application kernels such as matrix multiplication (K3)
CO4:	Write simple programs using OpenCL (K3)
CO5:	Write an efficient parallel program for a given problem(K3).

Part – A (6×2 = 12 Marks)

1	What are OpenCL Contexts?	K1	CO4	1.4.1 2.1.2
2	Identify the OpenCL API call used for finding the available platforms on the system and provide the corresponding syntax for this operation.	K3	CO4	1.4.1 2.2.2 13.3.1
3	What is the significance of OpenCL API call get_global_id(0)?	K1	CO4	1.4.1 2.1.2
4	Identify the OpenCL API call used for creating a memory object Buffer and provide the corresponding syntax for this operation.	K3	CO4	1.4.1 2.1.2
5	What is the difference between the exclusive and inclusive variants of prefixsum algorithm?	K1	CO5	1.3.1 13.3.1
6	Identify the OpenCL API call for creating a command-queue and include the relevant syntax for this action.	K3	CO4	1.4.1 2.2.2 13.3.1

Part – B (3×6 = 18 Marks)

7	Explain the key steps involved in running a basic OpenCL application.	K2	CO4	1.4.1 2.1.2
8	Explain the features of OpenCL.	K2	CO4	1.4.1 2.1.2
9	What are the key steps involved in image rotation using OpenCL?	K2	CO5	1.3.1 13.3.1

Part – C (2×10 = 20 Marks)

10	Create a vector addition program using OpenCL to efficiently compute the sum of two arrays on a GPU.	K3	CO4	1.4.1 2.2.2 13.3.1
	(OR)			
11	Implement a parallel histogram calculation using OpenCL, and what are some key considerations and challenges when optimizing the performance of your OpenCL histogram kernel for different GPU architectures?	K3	CO4	1.4.1 2.2.2 13.3.1
12	. // Develop an OpenCLprogram to perform matrix multiplication of two square matrices, A and B, both of size N x N.	K3	CO5	1.3.1 2.3.1 13.3.1 13.3.2
	(OR)			
13	Develop an OpenCL program for performing Prefix Sum on an array using parallel computing techniques.	K3	CO5	1.3.1 2.3.1 13.3.1 13.3.2

Prepared By

PAC Members

Approved By
(HOD/CSE)