Hall Ticket No | | | | | | | | | | | Question Paper Code: ACSB03

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

MODEL QUESTION PAPER – I

Four Year B.Tech III Semester End Examinations, November-2019

Regulations: R18

DATA STRUCTURES

(Common to ME/CSE/IT/ECE/CE)

Time: 3 hours **Max. Marks: 70**

Answer ONE Question from each Module

All Questions Carry Equal Marks

All parts of the question must be answered in one place only

MODULE – I

1. a) Define Data structure? Write the procedure for bubble sort using a suitable example? **[7M]**

   b) Write the name of the sorting technique which is used in playing cards game? Write a procedure for sorting a given list of numbers using that technique? **[7M]**
   14, 25, 36, 74, 85, 6, 53, 62, 41

2. a) Compare the time complexities of various searching and sorting algorithms? **[7M]**

   b) Given a list of integers 9, 12, 23, 30, 35, 42, 55, 61, 71, 82, 99. Search for an element 35 in the sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. **[7M]**

MODULE – II

3. a) Implement the basic stack operations PUSH, POP, DISPLAY using a list? **[7M]**

   b) Suppose a circular queue of capacity (n – 1) elements is implemented with an array of n elements. Assume that the insertion and deletion operation are carried out using REAR and FRONT as array index variables, respectively. Initially, REAR = FRONT = 0. Find the conditions to detect queue full and queue empty by using the following conditions. **[7M]**
      i. Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT
      ii. Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
      iii. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
      iv. Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT

4. a) Write a program to reverse a stack using recursion. Use the following ADT functions on Stack S: **[7M]**
   isEmpty(S)
   push(S)
   pop(S)

b) Design a data structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. Consider the following SpecialStack and return the minimum element in the current stack. **[7M]**

16 --> TOP
15
29
19
18

MODULE – III

5. a) Given a doubly linked list, write a function to sort the doubly linked list in increasing order using any sorting technique. **[7M]**

b) Given a singly linked list and a position, Write a program to delete a linked list node at the given position. **[7M]**

Input: position = 1, Linked List = 8->2->3->1->7 Output: Linked List = 8->3->1->7

Input: position = 0, Linked List = 8->2->3->1->7 Output: Linked List = 2->3->1->7

6. a) Write a function to implement the basic operations of a doubly linked list. **[7M]**

b) Given a singly linked list, write a program to check if the linked list is circular or not. A linked list is called circular if it not NULL terminated and all nodes are connected in the form of a cycle. **[7M]**

MODULE – IV

7. a) Describe various Non linear data structures and explain different techniques to represent Trees and Graphs. **[7M]**

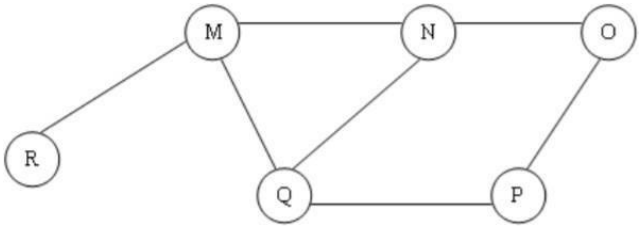
b) Construct a binary tree given the pre-order traversal and in-order traversals as follows: **[7M]**

i) Pre-Order Traversal: G B Q A C K F P D E R H
ii) In-Order Traversal: Q B K C F A G P E D H R

8. a) Construct a binary search tree by inserting following nodes in sequence: 68, 85, 23, 38, 44, 80, 30, 108, 26, 5, 92, 60. **[7M]**

Write in-order, pre-order and post-order traversal of the above generated Binary search tree.

b) The Breadth First Search algorithm has been implemented using the queue data structure. Discover breadth first search for the graph shown in Figure with starting node M. **[7M]**



MODULE – V

9. a) Write how an AVL tree is different from Binary search tree and Create a AVL tree and binary search tree for the given data: 56, 45,91,82,34,22,100,71,85,12 **[7M]**

   b) Create a B-Tree of order 4 for the following data: 67,33,57,81,20,11,16,38,61,78 **[7M]**

10. a) Define Hashing? Explain various collision resolution techniques?

    b) A hash table contains 10 buckets and uses linear probing to resolve collisions. The key values are integers and hash function used is (key mod 10). If the values 43, 165, 62, 123, 142 are inserted in the table, then find the location of the key value 142 in the table? **[7M]**

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
Dundigal, Hyderabad - 500 043

COURSE OBJECTIVES
The course should enable the students to:

I	Learn the basic techniques of algorithm analysis.
II	Demonstrate searching and sorting algorithms and analyse their time complexities.
III	Implement linear data structures viz. stack, queue and linked list.
IV	Demonstrate non-linear data structures viz. tree and graph traversal algorithms.
V	Study and choose appropriate data structure to solve problems in real world.

COURSE OUTCOMES (COs):

CO 1	Understand the concept of data structures and apply algorithm for solving problems like sorting, searching, insertion and deletion of data.
CO 2	Understand linear data structures for processing of ordered or unordered data.
CO 3	Explore various operations on dynamic data structures like single linked list, circular linked list and doubly linked list.
CO 4	Explore the concept of non linear data structures such as trees and graphs.
CO 5	Understand the binary search trees, hash function, and concepts of collision and its resolution methods.

COURSE LEARNING OUTCOMES (CLOs):

ACSB03.01	Understand algorithms and data structures in terms of time and space complexity of basic operations.
ACSB03.02	Choose a suitable algorithm to organize the data in ascending or descending order.
ACSB03.03	Explore an algorithm to find the location of an element in a given list.
ACSB03.04	Compare the time complexities of various searching and sorting algorithms.
ACSB03.05	Implementation of stack and queues using an underlying array.
ACSB03.06	Understand application of stacks in arithmetic expression conversion and evaluation.
ACSB03.07	Understand working of circular queues and double ended queue.
ACSB03.08	Understand dynamic data structures and their real time applications.
ACSB03.09	Understand the basic insertion and deletion operations associated with linked list.
ACSB03.10	Organize the data in various linked representation format.
ACSB03.11	Understand the concept of non-linear data structures viz. trees and graphs.
ACSB03.12	Application of trees, graphs and graph traversal techniques.
ACSB03.13	Compare and Contrast the operations of binary search trees and AVL trees.
ACSB03.14	Understand the concept of M-way search trees, operations and applications.
ACSB03.15	Understand the implementation of hashing using hash table and hash function.
ACSB03.16	Describe the concept of collision and its resolving methods in applications.
ACSB03.17	Strengthen the knowledge of data structures and algorithms for employability.

MAPPING OF SEMESTER END EXAM TO COURSE LEARNINIG OUTCOMES

SEE Question No		Course Learning Outcomes		Course Outcomes	Blooms Taxonomy Level
1	a	ACSB03.01	Understand algorithms and data structures in terms of time and space complexity of basic operations.	CO 1	Remember
	b	ACSB03.02	Choose a suitable algorithm to organize the data in ascending or descending order.	CO 1	Understand
2	a	ACSB03.04	Compare the time complexities of various searching and sorting algorithms.	CO 1	Understand
	b	ACSB03.03	Explore an algorithm to find the location of an element in a given list.	CO 1	Understand
3	a	ACSB03.05	Implementation of stack and queues using an underlying array.	CO 2	Remember
	b	ACSB03.07	Understand working of circular queues and double ended queue.	CO 2	Understand
4	a	ACSB03.05	Implementation of stack and queues using an underlying array.	CO 2	Understand
	b	ACSB03.05	Implementation of stack and queues using an underlying array.	CO 2	Understand
5	a	ACSB03.08	Understand dynamic data structures and their real time applications.	CO 3	Remember
	b	ACSB03.09	Understand the basic insertion and deletion operations associated with linked list.	CO 3	Understand
6	a	ACSB03.10	Organize the data in various linked representation format.	CO 3	Remember
	b	ACSB03.09	Understand the basic insertion and deletion operations associated with linked list.	CO 3	Understand
7	a	ACSB03.11	Understand the concept of non-linear data structures viz. trees and graphs.	CO 4	Understand
	b	ACSB03.12	Application of trees, graphs and graph traversal techniques.	CO 4	Apply
8	a	ACSB03.12	Application of trees, graphs and graph traversal techniques.	CO 4	Understand
	b	ACSB03.12	Application of trees, graphs and graph traversal techniques.	CO 4	Understand
9	a	ACSB03.13	Compare and Contrast the operations of binary search trees and AVL trees.	CO 5	Remember
	b	ACSB03.13	Compare and Contrast the operations of binary search trees and AVL trees.	CO 5	Understand
10	a	ACSB03.16	Describe the concept of collision and its resolving methods in applications.	CO 5	Remember
	b	ACSB03.15	Understand the implementation of hashing using hash table and hash function.	CO 5	Understand

Signature of Course Coordinator **HOD, CSE**