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

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

MODEL QUESTION PAPER-I

B.Tech III Semester End Examinations, November 2020

Regulations: IARE - R18

DATA STRUCTURES

(COMMON TO ME/CSE/IT/ECE/CE)

Time: 3 hour Maximum 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 sorting? Write the procedure for bubble sort using a suitable example? [**7m**]
   (b) Write a selection sort algorithm and also discuss its efficiency? [**7m**]

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**]
      1. Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT
      2. Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR
      3. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT
      4. 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 [**7m**]
      (i) on Stack (S)
      (ii) isEmpty(S)
      (iii) push(S)

(iv) 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. 16−− > TOP 15 29 19 18 [**7m**]

MODULE-III

5. (a) Given a doubly linked list, write a function to sort the doubly linked list in increasing order using merge sort. [**7m**]

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

(i) Input: position = 1, Linked List = 8− >2− >3− >1− >7
Output: Linked List = 8− >3− >1− >7

(ii) 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 collision resolution techniques in hashing. [**7m**]

(b) Construct a binary tree given the pre-order traversal and in-order traversals as follows:

(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 [**7m**]

8. (a) Write the procedure to be followed during infix to postfix conversion. Convert the following expression $(2+3)-(4/5)^7$ from infix to postfix form. [**7m**]

(b) The Breadth First Search algorithm has been implemented using the queue data structure. Discover breadth first search for the graph shown in **fig 1** with starting node M. [**7m**]

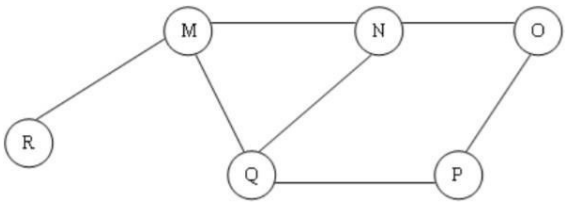


Figure 1: 8B

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? [**7m**]

(b) Define a binary tree? Construct a binary tree given the pre-order traversal and in-order traversals as follows:

(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

****END OF EXAMINATION****

COURSE OBJECTIVES:

The course should enable the students to:

1	To provide students with skills needed to understand and analyze performance trade-offs of different algorithms/implementations and asymptotic analysis of their running time and memory usage.
2	To provide knowledge of basic abstract data types (ADT) and associated algorithms: stacks, queues, lists, tree, graphs, hashing and sorting, selection and searching.
3	The fundamentals of how to store, retrieve, and process data efficiently.
4	To provide practice by specifying and implementing these data structures and algorithms in Python.
5	Understand essential for future programming and software engineering courses.

COURSE OUTCOMES:

After successful completion of the course, students should be able to:

CO 1	Carryout the analysis of a range of algorithms in terms of algorithm analysis and express algorithm complexity using the O notation.
CO 2	Make use of recursive algorithm design technique in appropriate contexts.
CO 3	Represent standard ADTs by means of appropriate data structures.
CO 4	Select appropriate sorting technique for given problem.
CO 5	Select appropriate searching technique for given problem.
CO 6	Implement standard searching and sorting algorithms; including binary search; merge sort and quick sort; and their complexities.
CO 7	Design and implement linked lists, stacks and queues in Python.
CO 8	Explain the use of basic data structures such as arrays, stacks, queues and linked lists in program design.
CO 9	Extend their knowledge of data structures to more sophisticated data structures to solve problems involving balanced binary search trees, AVL Trees, B-trees and B+ trees, hashing, and basic graphs.
CO 10	Design and implement tree structures in Python.
CO 11	Compare and contrast the benefits of dynamic and static data structures implementations and choose appropriate data structure for specified problem domain.
CO 12	Quickly determine and explain how efficient an algorithm or data structure will be, apply appropriate data structures for solving computing problems with respect to performance.

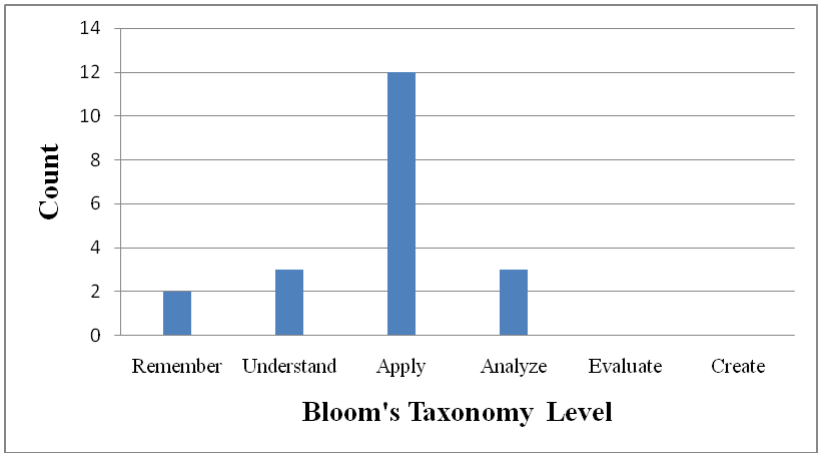
MAPPING OF SEMESTER END EXAMINATION QUESTIONS TO COURSE OUTCOMES

Q.No		All Questions carry equal marks	Taxonomy	CO's	PO's
1	a	Define sorting? Write the procedure for bubble sort using a suitable example?	Remember	CO 4,6,12	PO 1
	b	Write a selection sort algorithm and also discuss its efficiency?	Understand	CO 4,6,12	PO 1
2	a	Compare the time complexities of various searching and sorting algorithms?	Apply	CO 1	PO 1
	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.	Apply	CO 4,5,6,12	PO 1
3	a	Implement the basic stack operations PUSH, POP, DISPLAY using a list?	Apply	CO 7,8,11,12	PO 1,2,3
	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. 1. Full: (REAR+1) mod n == FRONT, empty: REAR == FRONT 2. Full: (REAR+1) mod n == FRONT, empty: (FRONT+1) mod n == REAR 3. Full: REAR == FRONT, empty: (REAR+1) mod n == FRONT 4. Full: (FRONT+1) mod n == REAR, empty: REAR == FRONT	Analyze	CO 7,8,11,12	PO 1,2,3
4	a	Write a program to reverse a stack using recursion. Use the following ADT functions (i) on Stack (S) (ii) isEmpty(S) (iii) push(S) (iv) pop(S).	Apply	CO 7,8,11,12	PO 1,2,3

	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. 16−− > TOP 15 29 19 18	Apply	CO 7,8,11,12	PO 1,2,3
5	a	Given a doubly linked list, write a function to sort the doubly linked list in increasing order using merge sort.	Apply	CO 7,8,11,12	PO 1,2,3
	b	Given a singly linked list and a position, Write a program to delete a linked list node at the given position. (i) Input: position = 1, Linked List = 8− >2− >3− >1− >7 Output: Linked List = 8− >3− >1− >7 (ii) Input: position = 0, Linked List = 8− >2− >3− >1− >7 Output: Linked List = 2− >3− >1− >7	Apply	CO 7,8,11,12	PO 1,2,3
6	a	Write a function to implement the basic operations of a doubly linked list.	Understand	CO 7,8,11,12	PO 1,2,3
	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.	Analyze	CO 7,8,11,12	PO 1,2,3
7	a	Describe various collision resolution techniques in hashing.	Remember	CO 9,12	PO 1,2,3
	b	Construct a binary tree given the pre-order traversal and in-order traversals as follows: (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	Analyze	CO 8,9,10,11	PO 1,2,3
8	a	Write the procedure to be followed during infix to postfix conversion. Convert the following expression $(2+3)-(4/5)^7$ from infix to postfix form.	Apply	CO 7,8,11,12	PO 1,2,3
	b	The Breadth First Search algorithm has been implemented using the queue data structure. Discover breadth first search for the graph shown in **fig 1** with starting node M.	Apply	CO 7,8,11,12	PO 1,2,3

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	Understand	CO 8,9,10,11	PO 1,2,3
	b	Create a B-Tree of order 4 for the following data: $67, 33, 57, 81, 20, 11, 16, 38, 61, 78$.	Apply	CO 8,9,10,11	PO 1,2,3
10	a	Define Hashing? Explain various collision resolution techniques?.	Understand	CO 9,12	PO 1,2,3
	b	Define a binary tree? Construct a binary tree given the pre-order traversal and in-order traversals as follows: (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	Apply	CO 8,9,10,11	PO 1,2,3

KNOWLEDGE COMPETENCY LEVELS OF MODEL QUESTION PAPER



1.png

Signature of Course Coordinator **HOD, ME**
Ms. B Padmaja, Assistant Professor, CSE