REG.NO.:

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**
**CONTINUOUS ASSESSMENT TEST - II**
**FALL SEMESTER 2024-2025**

**SLOT: B2+TB2**

---

**Programme Name & Branch** : **B.Tech & Computer Science and Engineering**
**Course Code and Course Name** : **BCSE202L & Data Structures and Algorithms**
**Faculty Name(s)** : **ALL**
**Class Number(s)** : **Common to all batches**
**Date of Examination** : **14-Oct-2024**
**Exam Duration** : **90 minutes** **Maximum Marks: 50**

**General instruction(s):**

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes
  2. Articulate linear, non-linear data structures and legal operations permitted on them.
  3. Identify and apply suitable algorithms for searching and sorting.
  4. Discover various tree and graph traversals.
  5. Explicate hashing, heaps and AVL trees and realize their applications

<table>
<tr><th>Q. No</th><th>Question</th><th>M</th><th>CO</th><th>BL</th></tr>
<tr><td>1.</td><td>a) Illustrate the working of quicksort on the array A = [14, 13, 16, 19, 87, 14, 32, 11, 12, 21, 18, 39, 32]. Assume that the first element is chosen as the pivot when the algorithm partitions the array. Show the array after every partition and trace the recursive calls made by the algorithm. (8 Marks)<br>Ans:<br>QuickSort (arr, 0 to 12) 14 13 16 19 87 14 32 11 12 21 18 39 32<br>QuickSort (arr, 0 to 2) 11 13 12<br>QuickSort (arr, 1 to 2) 13 12<br>QuickSort (arr, 4 to 12) 16 19 87 14 32 21 18 39 32<br>QuickSort (arr, 6 to 12) 19 87 32 21 18 39 32<br>QuickSort (arr, 8 to 12) 87 32 21 39 32<br>QuickSort (arr, 8 to 11) 32 32 21 39<br>QuickSort (arr, 10 to 11) 32 39<br>Sorted array: 11 12 13 14 14 16 18 19 21 32 32 39 87<br><br>b) Following are some of the properties of sorting algorithms. Match the property (characteristics) against the sorting algorithm it corresponds to. Note: Matching can be of any type. One-to-many, many-to-one, one-to-one, etc. (2 Marks)<br><table><tr><th>Properties</th><th>Sorting algorithms</th></tr><tr><td>a. It uses divide-and-conquer principle and sorts the two subsequences of sorted elements into one.<br>b. For each input element x, it identifies the number of elements greater than x and fixes the position for x (when the output of the elements is in descending order).</td><td>1. Quicksort<br>2. Merge sort<br>3. Counting sort<br>4. Insertion sort<br>5. Selection sort</td></tr></table>Ans:<br>a. Merge Sort<br>b. Counting Sort</td><td>10</td><td>3</td><td>4</td></tr>
</table>

REG.NO.:

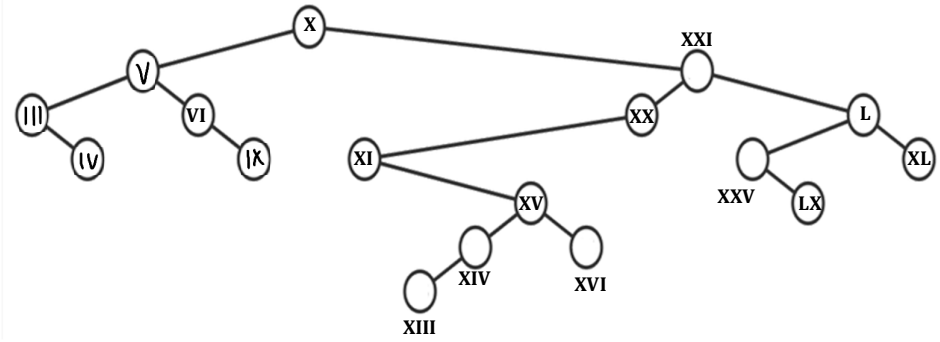
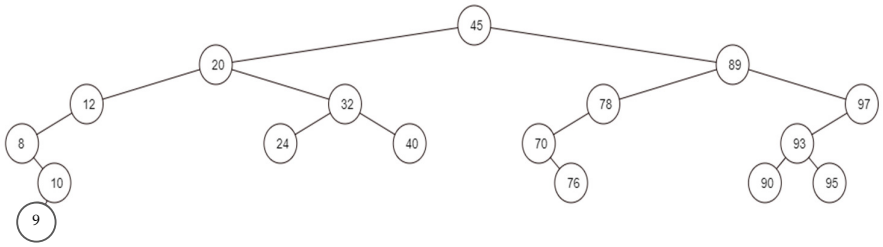
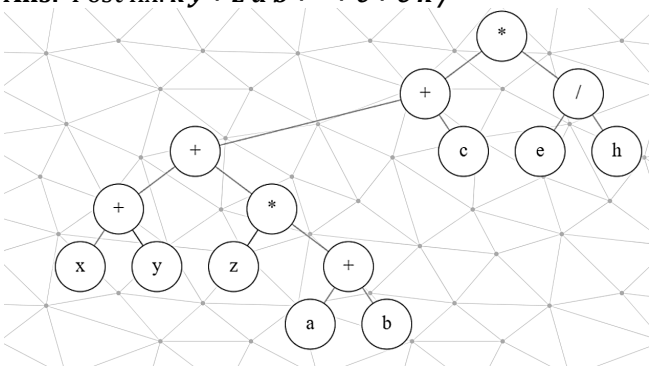
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**
**CONTINUOUS ASSESSMENT TEST - II**
**FALL SEMESTER 2024-2025**

**SLOT: B2+TB2**

---

<table>
<tr>
<td>2.</td>
<td>
a. Explain two scenarios in which (i) array is preferred over linked list (ii) Linked list is preferred over array. Write your answer with the following format (portrait or landscape). Your answer is legible, concise and draw diagrams if needed. (8M)<br>
Ans: Any two for each case with necessary examples related to memory and ease of access.<br><br>
(i) Array is better over Linked list for:
<table>
<tr><th>Sl. No.</th><th>Scenario / Operation</th><th>(what happens for) Array</th><th>(what happens for) Linked List</th></tr>
<tr><td>1</td><td>Access A[i] --- Probe</td><td>Direct Access. O(1) time for array</td><td>not possible (for linked list)</td></tr>
<tr><td>2.</td><td>Insert at the end as last element</td><td>No shuffling</td><td></td></tr>
</table>
(ii) Linked list is better over array for:
<table>
<tr><th>Sl. No.</th><th>Scenario / Operation</th><th>Linked list</th><th>Array</th></tr>
<tr><td>1</td><td>Insert at front</td><td>no shuffle, just change pointers</td><td>Shuffle required.</td></tr>
<tr><td>2.</td><td>Delete at the tail.</td><td>no shuffle, just change pointers</td><td>Shuffle required.</td></tr>
<tr><td>3.</td><td>Memory management/allotment</td><td>memory management is efficient</td><td>memory management is not efficient</td></tr>
</table>
b. Consider the following code snippet where x may be a data value in a SLL).<br>
struct node * q = head; // q stores the address of head of a singly L.L<br>
while (q != null && q → data !=x)<br>
{ q = q → next;<br>
}<br>
return q;<br>
Question: If (a) q==null<br>
(b) q != null,<br>
what do you conclude? (2 marks)<br>
Ans<br>
(a) If q is null, list is empty or element x is not in the list.<br>
(b) If q is not null, it will return the node having the value x.
</td>
<td>10</td><td>2</td><td>5</td>
</tr>
<tr>
<td>3.</td>
<td>
Insert the following Roman numerals into a binary search tree.<br>
X, XXI, V, VI, III, IX, L, XX, XXV, XI, XV, XIV, IV, XVI, XIII, XL, LX<br>
From the above-created BST, remove the root node. Illustrate the resultant tree.<br>
(The general convention is that V is 5, X is 10, and L is 50).
</td>
<td>10</td><td>4</td><td>3</td>
</tr>
</table>

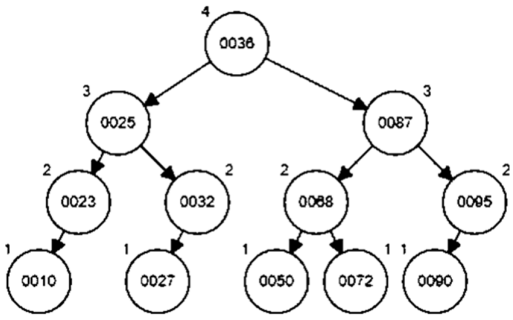
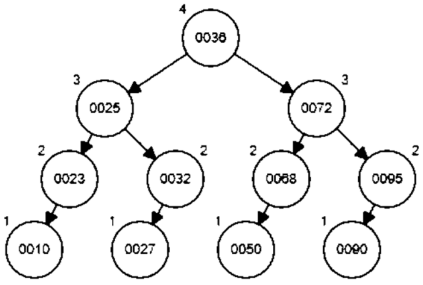
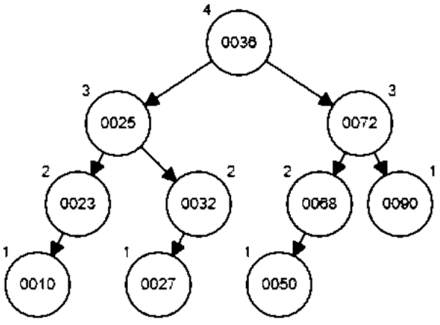
| | | | | |
|---|---|---|---|---|
| |  | | | |
| 4. | a) Given the following preorder traversal of the binary search tree, create the tree: 45 20 12 8 10 9 32 24 40 89 78 70 76 97 93 90 95<br>Is it possible to construct the binary tree (not binary search tree) from preorder and postorder traversals? Justify with suitable example(s).<br>**Ans**<br>  <br>*Not possible. For example, consider given Preorder traversal: AB*<br>*Postorder traversal: BA*<br>  <br>*Given only preorder and postorder traversal, it is not possible to create the tree.* | 6 | 4 | 4 |
| | b) Draw the expression tree for the given expression:<br>***((x + y) + z * (a + b) + c) * (e / h)***<br>Perform post-order traversal of the tree.<br>**Ans:** Post fix: ***x y + z a b + * + c + e h / ****<br>  | 4 | | |

REG.NO.:

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**
**CONTINUOUS ASSESSMENT TEST - II**
**FALL SEMESTER 2024-2025**

**SLOT: B2+TB2**

| | | | | |
|---|---|---|---|---|
| 5. | Demonstrate what happens when you insert the following sequence of numbers into an AVL tree: 50, 68, 95, 72, 87, 36, 23, 10, 25, 32, 27, 90. The balance factor for every node must be calculated. After every rotation, the new tree must be shown. In the resultant tree, delete node 87, followed by 95.<br><br>**Ans**<br>  <br><br>After deleting 87 (taking max from left subtree)<br>  <br><br>After deleting 95 (case 2)<br>  <br> | 10 | 5 | 3 |