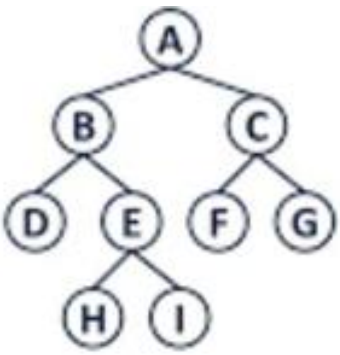
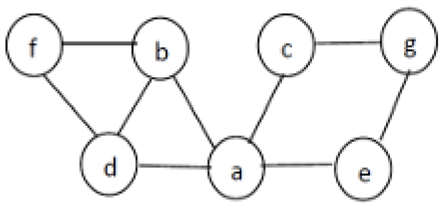
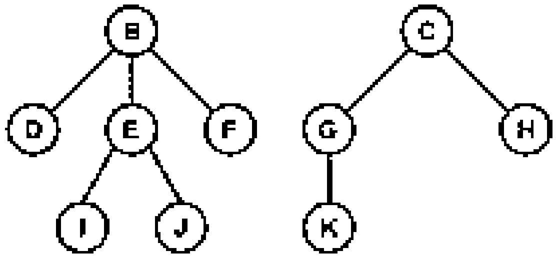
USN

Internal Assessment Test 2 – Jan 2026

Sub:	Data Structures and Applications				Sub Code:	BCS304	Branch:	CSE
Date:	8-1-2026	Duration:	90 mins	Max Marks: 50	Sem / Sec:	III (A,B &C)		**OBE**

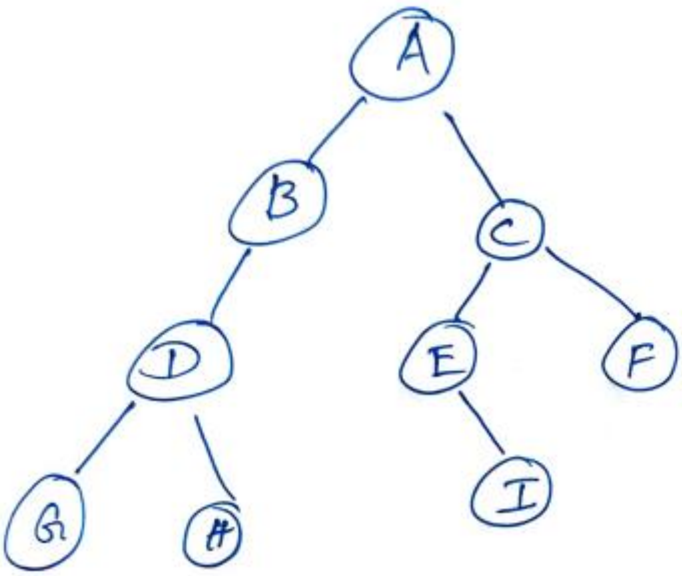
	Answer any FIVE FULL Questions	**MARKS**	**CO**	**RBT**
1.a.	Construct a binary tree from the Post-order and In-order sequence given below In-order: GDHBAEICF Post-order: GHDBIEFC A	**5**	**4**	**L3**
1 b.	Define Binary Search tree. Construct a binary search tree (BST) for the following elements: 100, 85, 45, 55, 120, 20, 70, 90, 115, 65, 130, 145.	**5**	**4**	**L2**
2. a.	Discuss Inorder, Preorder and Postorder traversal with suitable C- function for each.	**5**	**4**	**L2**
2.b	Find inorder, preorder, postorder and levelorder traversals for the given tree. 	**5**	**4**	**L2**
3.a.	Define the following with respect to a graph with suitable example: a. Connected Graph b. Directed Graph c. Multigraph d. Complete Graph e. Subgraph	**5**	**4**	**L1**
3.b.	Design an algorithm to traverse a graph using Depth First Search (DFS). Apply DFS for the graph given below starting at **'f'.** 	**5**	**4**	**L2**

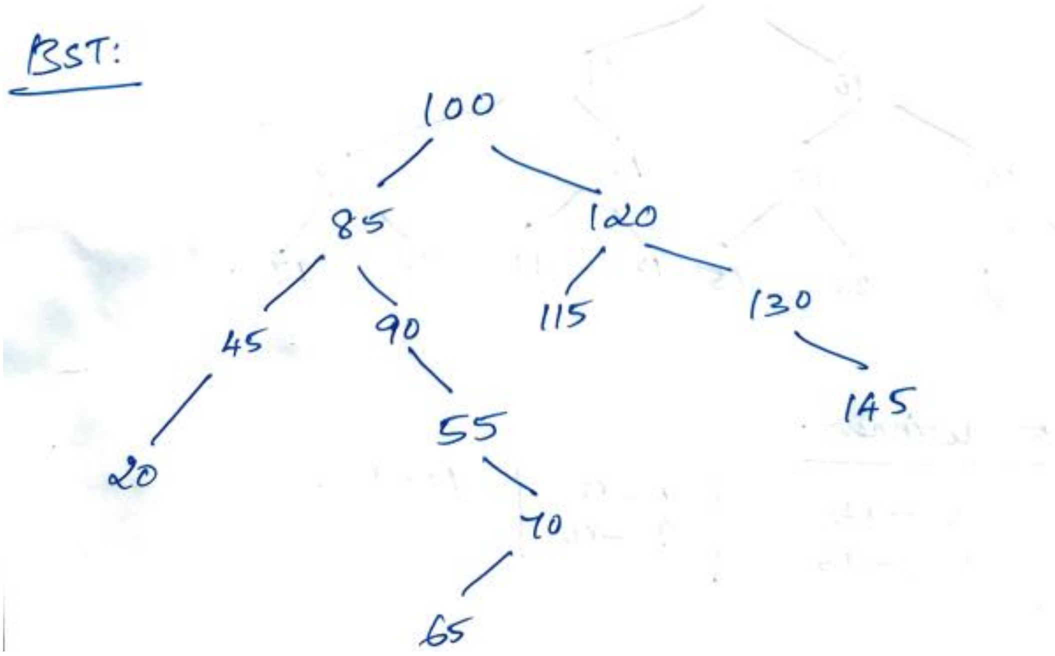
<table>
<tr><td>4.a.</td><td>Define Forest. Transform the given forest into a Binary tree.

</td><td>5</td><td>4</td><td>L2</td></tr>
<tr><td>4.b.</td><td>Define selection tree. Construct winner tree for the runs of a game given below. Each run consists of values of players. Find the first 5 winners.
<table>
<tr><td>10</td><td>9</td><td>20</td><td>6</td><td>8</td><td>9</td><td>90</td><td>17</td></tr>
<tr><td>15</td><td>20</td><td>20</td><td>15</td><td>15</td><td>11</td><td>95</td><td>18</td></tr>
<tr><td>16</td><td>38</td><td>30</td><td>25</td><td>50</td><td>16</td><td>99</td><td>20</td></tr>
<tr><td></td><td></td><td></td><td>28</td><td></td><td></td><td></td><td></td></tr>
</table></td><td>5</td><td>4</td><td>L2</td></tr>
<tr><td>5.a.</td><td>Write short notes on priority queues with suitable examples.</td><td>5</td><td>5</td><td>L1</td></tr>
<tr><td>5.b.</td><td>Construct an OBST given keys (a1,a2,a3) = (5,10,15) with equal probabilities pi=qi=1/7.</td><td>5</td><td>5</td><td>L2</td></tr>
<tr><td>6.a.</td><td>What is a collision? Explain the method used to resolve collisions using a suitable algorithm for linear probing and Chaining. Insert the keys 72, 27, 36, 24, 63, 81, 92, and 101 into a hash table of size 10 using the remainder (modulo) method.</td><td>5</td><td>5</td><td>L2</td></tr>
<tr><td>6.b.</td><td>Explain Hashing with any three hash functions.</td><td>5</td><td>5</td><td>L3</td></tr>
</table>

CI **CCI** **HOD**

Internal Assessment Test 2 – Jan 2026 : Solution

Sub:	Data Structures and Applications					Sub Code:	BCS304	Branch:	CSE
Date:	8-1-2026	Duration:	90 mins	Max Marks:	50	Sem / Sec:	III (A, B & C)		**OBE**

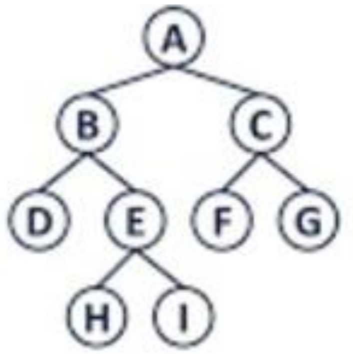
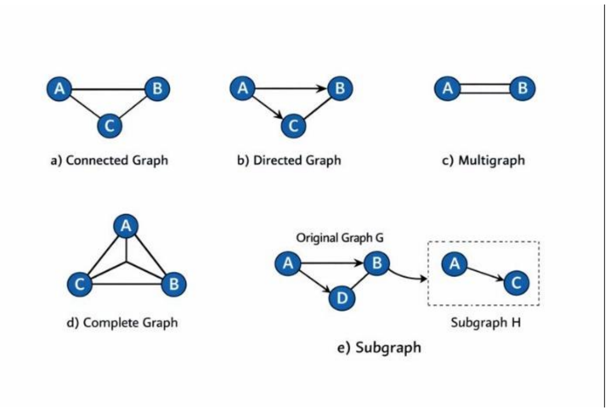
	Answer Key	**MARKS**	**CO**	**RBT**
1.a.	Construct a binary tree from the Post-order and In-order sequence given below In-order: GDHBAEICF Post-order: GHDBIEFCA  	**5**	**4**	**L3**
1 b.	Define Binary Search tree. Construct a binary search tree (BST) for the following elements: 100, 85, 45, 55, 120, 20, 70, 90, 115, 65, 130, 145.	**5**	**4**	**L2**



2. a.	Discuss Inorder, Preorder, Postorder and Level Order traversal with suitable C- function for each.	5	4	L2

```
void inorderTraversal(struct Node* root) {
    if (root != NULL) {
        // 1. Visit left subtree
        inorderTraversal(root->left);
        // 2. Visit root node
        printf("%d ", root->data);
        // 3. Visit right subtree
        inorderTraversal(root->right);
    }
}
void preorderTraversal(struct Node* root) {
    if (root != NULL) {
        // 1. Visit root node
        printf("%d ", root->data);
        // 2. Visit left subtree
        preorderTraversal(root->left);
        // 3. Visit right subtree
        preorderTraversal(root->right);
    }
}
void postorderTraversal(struct Node* root) {
    if (root != NULL) {
```

	// 1. Visit left subtree
	postorderTraversal(root->left);
	// 2. Visit right subtree
	postorderTraversal(root->right);
	// 3. Visit root node
	printf("%d ", root->data);
	}
	}			

2.b	Find all the traversals for the given tree  Inorder: DBHEIAFCG Preoder: ABDEHICFG Postorder: DHIEBFGCA Level Order: ABCDEFGHI	**5**	**4**	**L2**
3.a.	Define the following with respect to a graph with suitable example: a. Connected Graph b. Directed Graph c. Multigraph d. Complete Graph e. Subgraph  	**5**	**4**	**L1**

a. Connected Graph

Definition:

A graph is called **connected** if there is a **path between every pair of vertices**.

Example:

Vertices: {A, B, C}

Edges: {AB, BC}

Graph:

A — B — C

➡ Every vertex can be reached from any other vertex, so it is connected.

b. Directed Graph (Digraph)

Definition:

A **directed graph** is a graph in which edges have a **direction**, shown by arrows.

Example:

Vertices: {A, B, C}

Edges: {A→B, B→C}

Graph:

A → B → C

➡ Movement is only allowed in the direction of the arrows.

c. Multigraph

Definition:

A **multigraph** is a graph that can have **multiple edges between the same pair of vertices**.

Example:

Vertices: {A, B}

Edges: Two edges between A and B

Graph:

A == B

➡ More than one edge connects the same vertices.

d. Complete Graph

Definition:

A graph is called **complete** if **every pair of distinct vertices is connected by an edge**.

Example:

A complete graph with 3 vertices (K_3):

Vertices: {A, B, C}

Edges: {AB, BC, AC}

Graph:

A — B

\ /

C

➡ Every vertex is connected to all others.

e. Subgraph

Definition:

A **subgraph** is a graph formed from a **subset of vertices and edges** of a given graph.

Example:

Original Graph G:

Vertices: {A, B, C, D}

Edges: {AB, BC, CD}

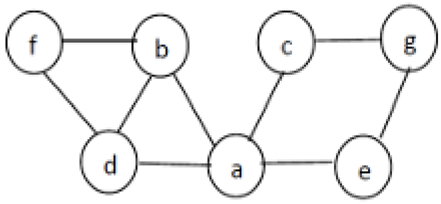
Subgraph H:

Vertices: {A, B, C}

Edges: {AB, BC}

➡ H is part of G, so it is a subgraph.

3.b.	Design an algorithm to traverse a graph using Depth First Search (DFS). Apply DFS for the graph given below starting at **'f'.**	**5**	**4**	**L2**

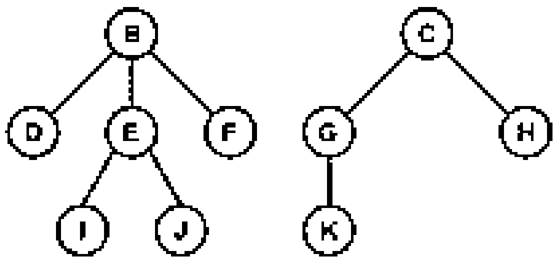



```
void dfs(int start,int n)
{
int visited[n]={0};
int stack[MAX];
int top=-1,i;
printf("%d->",start);
visited[start]=1;
stack[++top]=start;
while(top!=-1)
{
        start=stack[top];
        for(i=0;i<MAX;i++)
        {
                if(adj[start][i] && visited[i]==0)
```

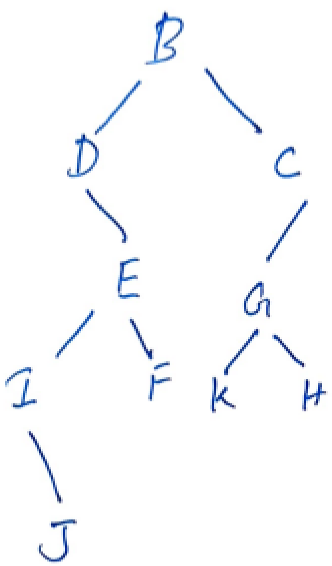
```
                    {
                    stack[++top]=i;
                    printf("%d->",i);
                    visited[i]=1;
                    break;
                    }
          }
          if(i==MAX)
                    top--;
}
}
```

DFS: f,b,a,c,g,e,d

4.a. Define Forest. Transform the given forest into a Binary tree. — **5** — **4** — **L2**



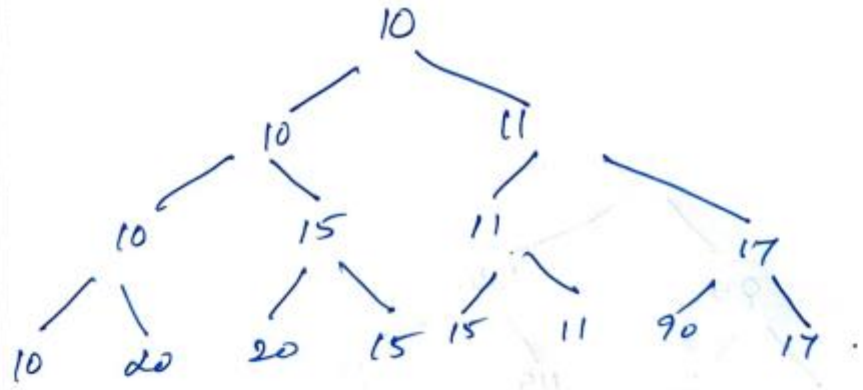
Solution:



4.b. Define selection tree. Construct winner tree for the runs of a game given below. Each run consists of values of players. Find the first 5 winners. — **5** — **4** — **L2**

10	9	20	6	8	9	90	17
15	20	20	15	15	11	95	18
16	38	30	25	50	16	99	20
			28				

Solution:

	 5 winner. 6 - r4 | 9 - r2 | 10 - r1. 8 - r5 | 9 - r6 |			
5.a.	Write short notes on priority queues with suitable examples. Priority Queues: A priority queue is a collection of elements such that element has an associated priority. There are two types of priority Queues. 1. Single-ended priority Queue. 2. Double-ended priority Queue.	**5**	**5**	**L1**

Heaps:

Heaps are frequently used to implement Priority Queues. In this Queue, the element to be deleted is the one with highest or lowest priority.

At any time, an element with an arbitrary priority can be inserted into the Queue

Max tree: A max tree is a tree in which the key value in each node is not smaller (larger) than the key values in its children.

A max heap is a complete binary tree that is also a max tree

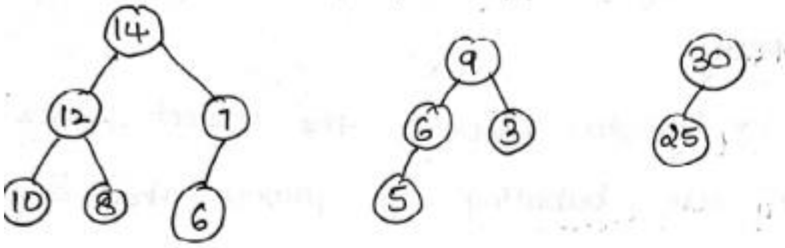
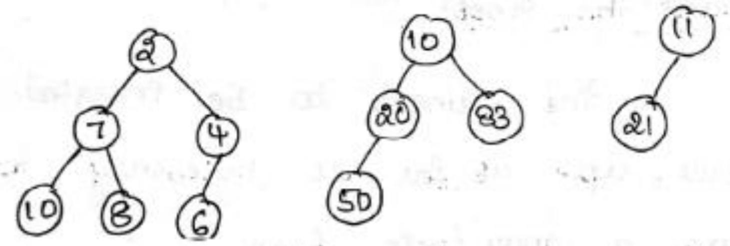
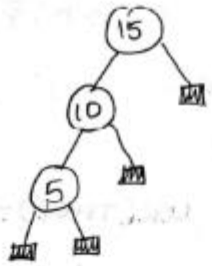
Min tree:

A min tree is a tree in which the key value in each node is not larger than the key value in its children.

A min heap is a complete binary tree that is also a min tree

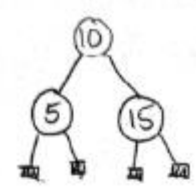
** Root of min tree is smallest.

** Root of max tree is largest.

	examples of max heaps:  examples of min heaps: 			
5.b.	Construct an OBST given keys (a1,a2,a3) = (5,10,15) with equal probabilities pi=qi=1/7.	**5**	**5**	**L2**
	sol: Given Keys 5, 10, 15 the possible distinct binary search trees are as follows: Level Tree a:  1 2 3 cost of tree a $Cost(tree\,a) = \frac{1}{7} + \frac{2}{7} + \frac{3}{7} + \frac{3}{7} + \frac{3}{7} + \frac{2}{7} + \frac{1}{7}$ $= \frac{16}{7}$			

tree b.

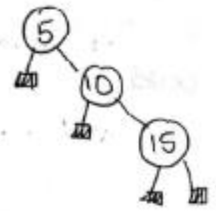
level
1
2



$$\text{cost(tree b)} = \frac{1}{7}+\frac{2}{7}+\frac{2}{7}+\frac{2}{7}+\frac{2}{7}+\frac{2}{7}+\frac{2}{7} = 13/7.$$

tree C

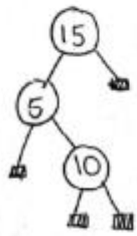
level
1
2
3



$$\text{Cost(TreeC)} = \frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{3}{7} = \frac{15}{7}$$

Tree D

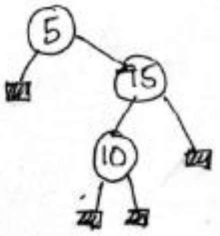
Level
1
2
3



$$\text{Cost(treeD)} = \frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{3}{7} = \frac{15}{7}$$

Tree E

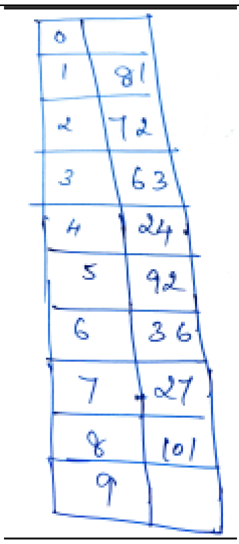
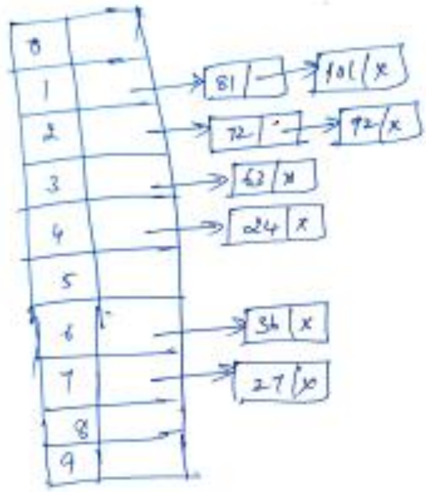
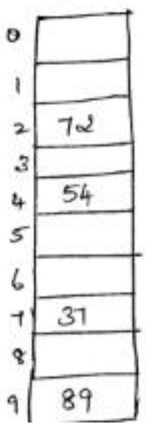
level
1
2
3



$$\text{cost [Tree E]} = \frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{1}{7}+\frac{2}{7}+\frac{3}{7}+\frac{3}{7} = \frac{15}{7}.$$

from the above Probabilities, tree b is optimal.

6.a.	What is a collision? Explain the method used to resolve collisions using a suitable algorithm for linear probing and Chaining. Insert the keys 72, 27, 36, 24, 63, 81, 92, and 101 into a hash table of size 10 using the remainder (modulo) method.	**5**	**5**	**L2**

	  **Linear Probing** **Chaining**			
6.b.	Explain Hashing with any three hash function. ① Division Method: * The hash function depends upon the remainder of division. * Typically the divisor is table length. Example: Record : 54, 72, 89, 37 h(key) : record % table size 	**5**	**5**	**L3**

2) Mid-Square :

In the mid square method, the key is squared and the middle or mid part of the result is used as the index.

Example: Record : 3111.

$3111^2 = 9678321$.

Hash table size 1000.

$H(3111) = 783$ (middle 3 - digits).

3) Digit Folding:

The key is divided into separate parts and using some simple operation these parts are combined to produce the hash key.

Example:

Record : 12365412

① Record divided into 123 654 12

② Added together $123+654+12$

$= 0789$.

$\Longrightarrow H(12365412) = 789$.

CI CCI HOD