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

School of Computer Science and Engineering

Fall Semester 2022-2023 Continuous Assessment Test – 2

Answer Key

SLOT B2+TB2

Programme Name & Branch: B.Tech. Computer Science and Engineering
Course Name & code: Data Structures and Algorithms **(BCSE202L)**

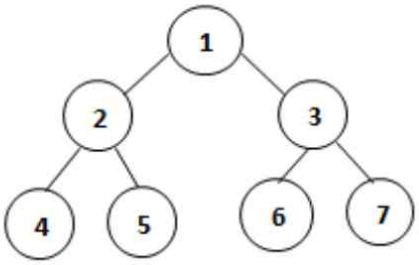
Exam Duration: 90 Min. **Maximum Marks: 50**

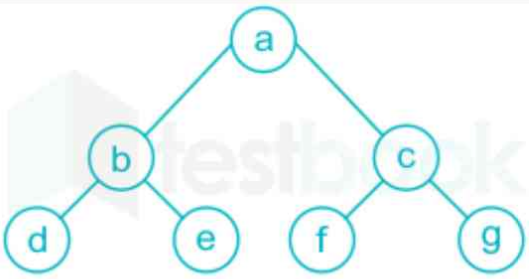
Answer ALL questions.

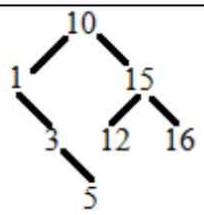
Q.No.	Question	Max Marks	CO	BL

1.	a) Compare singly, doubly and circular linked lists in detail. Write the recursive algorithm for reversing a singly linked list and discuss its time complexity. (6 Marks)
	
	Ans) Compare 3 popularly types of linked lists.
	
	Recursive algorithm:
	
	struct node *head;
	void reverse(struct node *prev, struct node *cur)
	{ if(cur)
	{ reverse(cur, cur->link);
	cur->link=prev;
	}
	else
	head=prev;
	}
	void main()
	{ reverse(NULL, head);
	}
	
	O(n) is the time complexity, where n is the size of the linked list.
	
	b) The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution? (4 Marks)
		10	CO2	BL2

	struct node
	{
	int value;
	struct node *next;
	};
	void rearrange(struct node *list)
	{
	struct node *p, * q;
	int temp;
	if ((!list) || !list->next)
	return;
	p = list;
	q = list->next;
	while(q)
	{
	temp = p->value;
	p->value = q->value;
	q->value = temp;
	p = q->next;
	q = p?p->next:0;
	}
	}
	
	Ans) 2,1,4,3,6,5,7			

2.	a) Consider the following tree:  If the post-order traversal gives ab-cd*+ then what will be the label of the nodes 1, 2, 3, …..? (4 Marks) **Ans)** +, -, *, a, b, c, d b) The inorder and preorder traversal of a binary tree is: d b e a f c g and a b d e c f g respectively. Show the step-by-step construction of a binary tree and write the post order traversal of it? (6 Marks) **Ans)** The tree is:	10	CO4	BL3

	 The post order traversal of the binary tree is: d e b f g c a			
3.	a) A binary tree T has 20 leaves. The no. of nodes in T having 2 children is? (4 Marks) **Ans)** The no. of nodes with 2 children is always one less than no. of leaves. So, the answer is 19. <u>Explanation:</u> Sum of all degrees = 2 * |E|. Here considering tree as a k-ary tree : Sum of degrees of leaves + Sum of degrees for Internal Node except root + Root's degree = 2 * (No. of nodes - 1). Putting values of above terms, L + (I-1)*(k+1) + k = 2 * (L + I - 1) L + k*I - k + I -1 + k = 2*L + 2I - 2 L + K*I + I - 1 = 2*L + 2*I - 2 K*I + 1 - I = L (K-1)*I + 1 = L Given k = 2, L=20 ==> (2-1)*I + 1 = 20 ==> I = 19 ==> T has 19 internal nodes which are having two children. b) The following numbers are inserted into an empty binary search tree in the given order: 10, 1, 3, 5, 15, 12, 16. What is the height of the BST? (2 Marks) **Ans)** The BST is:	10	CO4	BL4



Answer is 3.

c) Assume the height of a tree with a single node is 0. The number of ways in which the numbers 1, 2, 3, 4, 5, 6, 7 can be inserted in an empty binary search tree, such that the resulting tree has height 6, is? (4 Marks)

Ans) Answer is 64

Explanation:

To get height 6, we need to put either 1 or 7 at root.

So count can be written as T(n) = 2*T(n-1) with T(1) = 1

7

/

[1..6]

1

\

[2..7]

Therefore count is $2^6 = 64$

Another Explanation:

Consider these cases,
1 2 3 4 5 6 7
1 2 3 4 5 7 6
1 7 6 5 4 3 2
1 7 6 5 4 2 3
7 6 5 4 3 2 1
7 6 5 4 3 1 2
7 1 2 3 4 5 6
7 1 2 3 4 6 5

For height 6, we have 2 choices. Either we select the root as 1 or 7.
Suppose we select 7.
Now, we have 6 nodes and remaining height = 5.
So, now we have 2 ways to select root for this sub-tree also.
Now, we keep on repeating the same procedure till remaining

	height = 1 For this last case also, we have 2 ways. Therefore, total number of ways = 2^6= 64			

4.	a) Explain the BUILD_MAX_HEAP(A) operation/algorithm on a given array A. Also, analyse its time complexity? (5 Marks)
	
	Ans) // C program for building Heap from Array. Time complexity is O(n). The code is only given for the sake of extracting important information on how to build max heap.
	
	#include <stdio.h>
	
	// To heapify a subtree rooted with node i which is
	// an index in arr[]. N is size of heap
	void swap(int *a, int *b)
	{
	int tmp = *a;
	*a = *b;
	*b = tmp;
	}
	
	void heapify(int arr[], int N, int i)
	{
	int largest = i; // Initialize largest as root
	int l = 2 * i + 1; // left = 2*i + 1
	int r = 2 * i + 2; // right = 2*i + 2
	
	// If left child is larger than root
	if (l < N && arr[l] > arr[largest])
	largest = l;
	
	// If right child is larger than largest so far
	if (r < N && arr[r] > arr[largest])
	largest = r;
	
	// If largest is not root
	if (largest != i) {
	swap(&arr[i], &arr[largest]);
	
	// Recursively heapify the affected sub-tree
	heapify(arr, N, largest);
	}
	}
	
	// Function to build a Max-Heap from the given array
	void buildHeap(int arr[], int N)
	{
		10	CO5	BL3

```
        // Index of last non-leaf node
        int startIdx = (N / 2) - 1;

        // Perform reverse level order traversal
        // from last non-leaf node and heapify
        // each node
        for (int i = startIdx; i >= 0; i--) {
                heapify(arr, N, i);
        }
}

// A utility function to print the array
// representation of Heap
void printHeap(int arr[], int N)
{
        printf("Array representation of Heap is:\n");

        for (int i = 0; i < N; ++i)
                printf("%d ",arr[i]);
        printf("\n");
}

// Driver's Code
int main()
{
        int arr[] = {1, 3, 5, 4, 6, 13, 10, 9, 8, 15, 17};

        int N = sizeof(arr) / sizeof(arr[0]);

        // Function call
        buildHeap(arr, N);
        printHeap(arr, N);

        return 0;
}
```

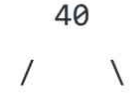
b) Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4. Now consider that a value 35 is inserted into this heap. After insertion, the new heap is? (2 Marks)

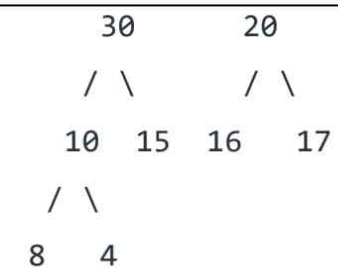
Ans) The new heap is:

40, 35, 20, 10, 30, 16, 17, 8, 4, 15

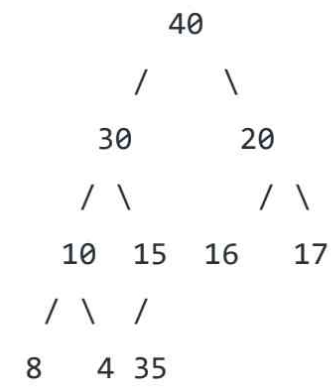
Explanation:

The array 40, 30, 20, 10, 15, 16, 17, 8, 4 represents following heap:

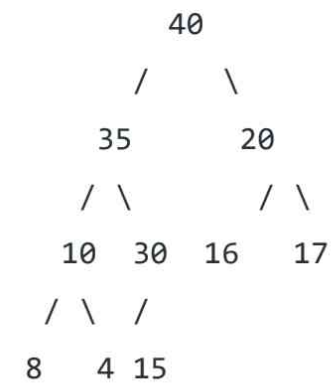




After insertion of 35, we get



After swapping 35 with 15 and swapping 35 again with 30, we get

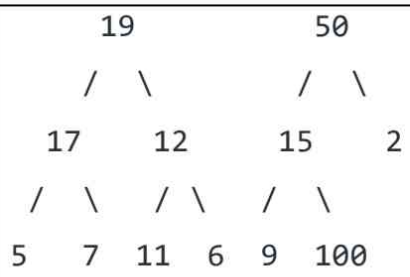
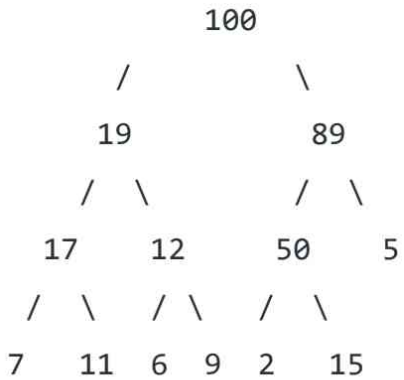


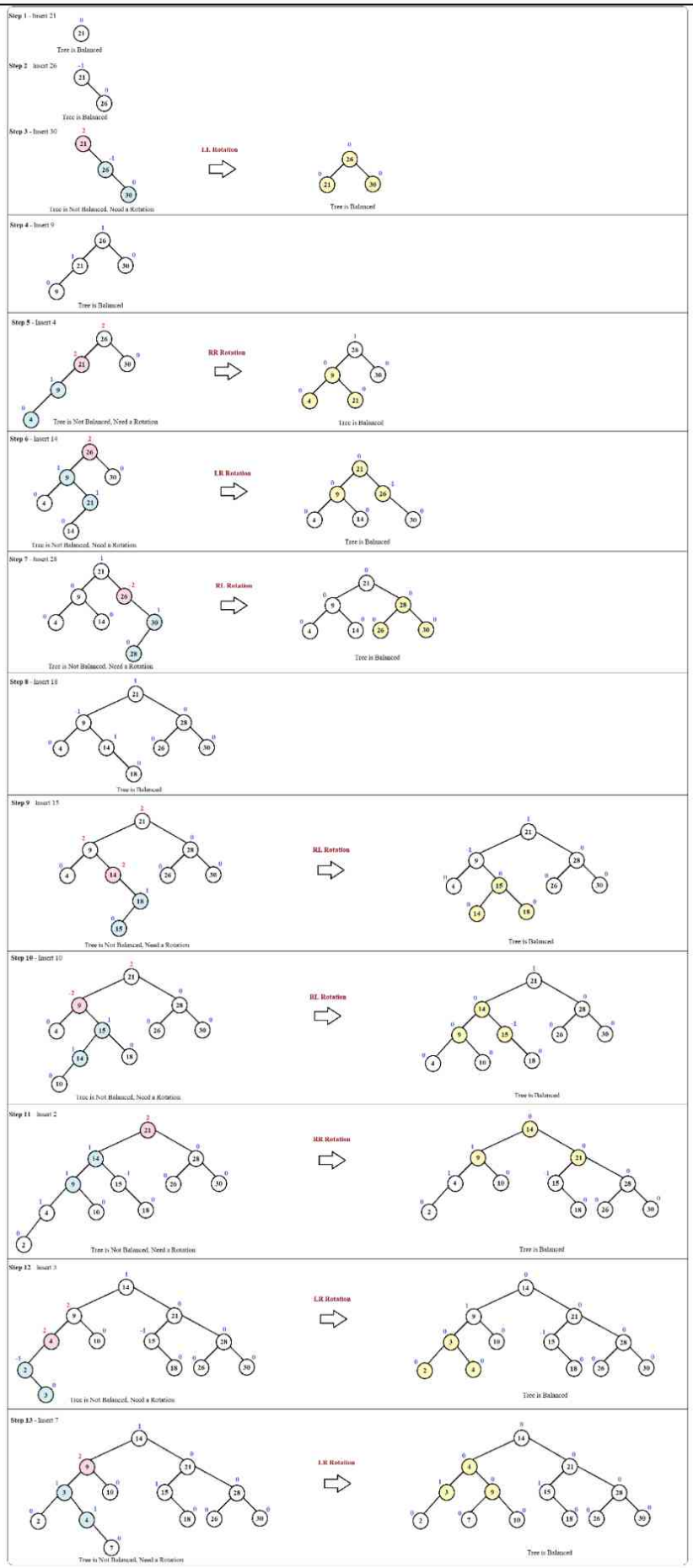
c) Consider the following array of elements. ⟨89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100⟩. The minimum number of interchanges needed to convert it into a max-heap is? (3 Marks)

Ans) Answer is 3.

Explanation:

⟨89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100⟩

	 Minimum number of swaps required to convert above tree to Max heap is 3. Below are 3 swap operations. Swap 100 with 15 Swap 100 with 50 Swap 100 with 89 			
5.	Show step-by-step construction of AVL tree for the following sequence: 21, 26, 30, 9, 4, 14, 28, 18, 15, 10, 2, 3, 7 Also, show how the deletion of elements 9 and 14 is performed in the AVL tree. (10 Marks) **Ans)**	10	CO5	BL5


Step 1 - Insert 21
Tree is Balanced
Step 2 - Insert 26
Tree is Balanced
Step 3 - Insert 30
Tree is Not Balanced, Need a Rotation
LL Rotation
Tree is Balanced
Step 4 - Insert 9
Tree is Balanced
Step 5 - Insert 4
Tree is Not Balanced, Need a Rotation
RR Rotation
Tree is Balanced
Step 6 - Insert 14
Tree is Not Balanced, Need a Rotation
LR Rotation
Tree is Balanced
Step 7 - Insert 28
Tree is Not Balanced, Need a Rotation
RL Rotation
Tree is Balanced
Step 8 - Insert 18
Tree is Balanced
Step 9 - Insert 15
Tree is Not Balanced, Need a Rotation
RL Rotation
Tree is Balanced
Step 10 - Insert 10
Tree is Not Balanced, Need a Rotation
RL Rotation
Tree is Balanced
Step 11 - Insert 2
Tree is Not Balanced, Need a Rotation
RR Rotation
Tree is Balanced
Step 12 - Insert 3
Tree is Not Balanced, Need a Rotation
LR Rotation
Tree is Balanced
Step 13 - Insert 7
Tree is Not Balanced, Need a Rotation
LR Rotation
Tree is Balanced

	Deletion in AVL tree is carried out using 2 steps: a) Deleting the element is same as in BST b) Rebalancing the imbalance node by performing rotations.			