INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SRI CITY

TERM-I EXAMINATION – SPRING 2024

Subject Name: ADSA

CSE:UG2(PC) **Date: 06-09-2024**

Duration: 90 Mins (3:30-5:00 PM) **Max. Marks: 25**

Instructions: **Roll No:**_____

1. All questions are compulsory.
2. Write the answers legibly.
3. Write Objective Type Questions Answers also in the Answer Sheet
4. Electronic Gadgets like mobile phones, laptops, smartwatches are not allowed.
5. Scientific Calculator is allowed

Section-A (Objective Type Questions)

1	Consider these functions: push() : push an element into the stack pop() : pop the top-of-the-stack element top() : returns the item stored in top-of-the-stack-node What will be the output after performing these sequence of operations push(20); push(4); top(); pop(); pop(); pop(); push(5); top(); **a.** 20 **b.** 4 **c.** 5 **d.** underflow	**[1 Mark]**
2	What is the correct recurrence relation for the time complexity of the Quicksort algorithm in the worst case? **a.** T(n)=T(2n)+O(n) **b.** T(n)=2T(2n)+O(n) **c.** T(n)=T(n−1)+O(n) **d.** T(n)=2T(n−1)+O(n)	**[1 Mark]**
3	Consider an array of *n* elements sorted in ascending order. You perform a Binary Search to find an element in this array. If the element is not present in the array, what is the maximum number of comparisons required to determine this? **a.** T(n)=T(2n)+O(n) **b.** T(n)=2T(2n)+O(n) **c.** T(n)=T(n−1)+O(n) **d.** T(n)=2T(n−1)+O(n)	**[1 Mark]**
4	Solve the recurrence relation T(n) = 2T (n/4) + n^0.51 **a.** Θ(n^0.51) **b.** Θ(n^0.50) **c.** O(n) **d.** O(sqrt(n))	**[1 Mark]**
5	What is the best case time complexity of radix sort algorithm **a.** **Ω**(nlog(n)) **b.** **Ω**(d(n+k))	**[1 Mark]**

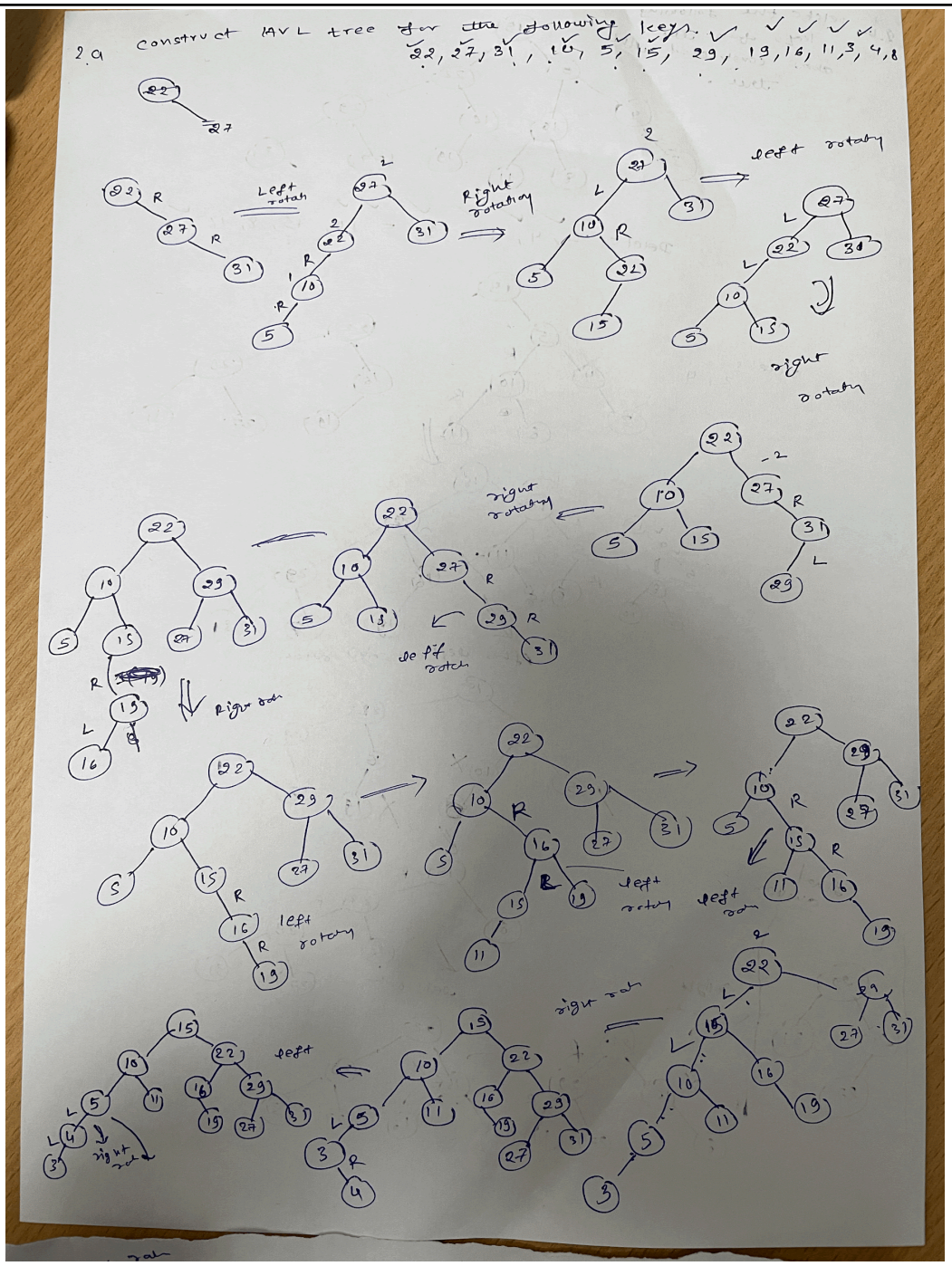
	c. **Ω**(dn) **d.** **Ω**(nk)	
6	What is the average case time complexity of Bucket Sort algorithm **a.** Θ(nlog(n)) **b.** Θ(d(n+k)) **c.** Θ(dn) **d.** Θ(n)	**[1 Mark]**
7	How many distinct binary search trees can be created out of 4 distinct keys? **a.** 4 **b.** 14 **c.** 24 **d.** 42	**[1 Mark]**
8	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 binary search tree (the height is the maximum distance of a leaf node from the root)? **a.** 2 **b.** 3 **c.** 4 **d.** 6	**[1 Mark]**
9	Which of the following sorting algorithm(s) is (are) stable **a.** heap sort **b.** counting sort **c.** quick sort **d.** selection sort	**[1 Mark]**
10	What is the purpose of using randomized quick sort over standard quick sort? **a.** so as to avoid worst case time complexity **b.** so as to avoid worst case space complexity **c.** to improve accuracy of output **d.** to improve average case time complexity	**[1 Mark]**

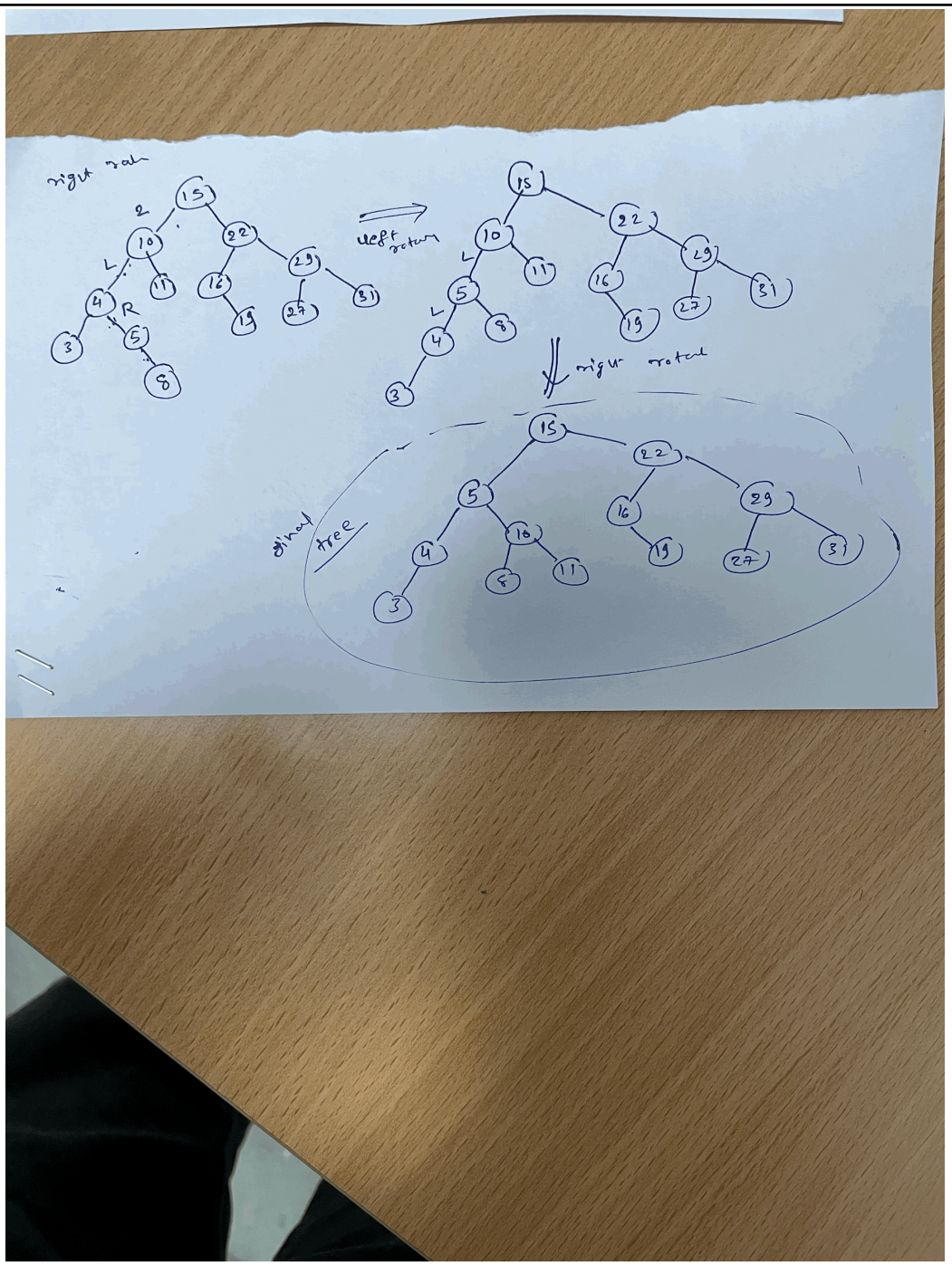
Section-B (Subjective Type Questions)

1.	Explain the Merge Sort algorithm through a structured algorithm (or pseudocode).
	`void mergeSort(int *Arr, int start, int end) {`
	`  if(start < end) {`
	`  int mid = (start + end) / 2;`
	`  mergeSort(Arr, start, mid);`
	`  mergeSort(Arr, mid+1, end);`
	`  merge(Arr, start, mid, end);`
	`  }`
	`}`
		[5 Marks]

```
void merge(int *Arr, int start, int mid, int end) {
        // create a temp array
        int temp[end - start + 1];
        // crawlers for both intervals and for temp
        int i = start, j = mid+1, k = 0;
        // traverse both arrays and in each iteration add smaller of both elements in temp
        while(i <= mid && j <= end) {
                if(Arr[i] <= Arr[j]) {
                        temp[k] = Arr[i];
                        k += 1; i += 1;
                }
                else {
                        temp[k] = Arr[j];
                        k += 1; j += 1;
                }
        }
        // add elements left in the first interval
        while(i <= mid) {
                temp[k] = Arr[i];
                k += 1; i += 1;}
        // add elements left in the second interval
        while(j <= end) {
                temp[k] = Arr[j];
                k += 1; j += 1; }
        // copy temp to original interval
        for(i = start; i <= end; i += 1) {
                Arr[i] = temp[i - start]}
}
```

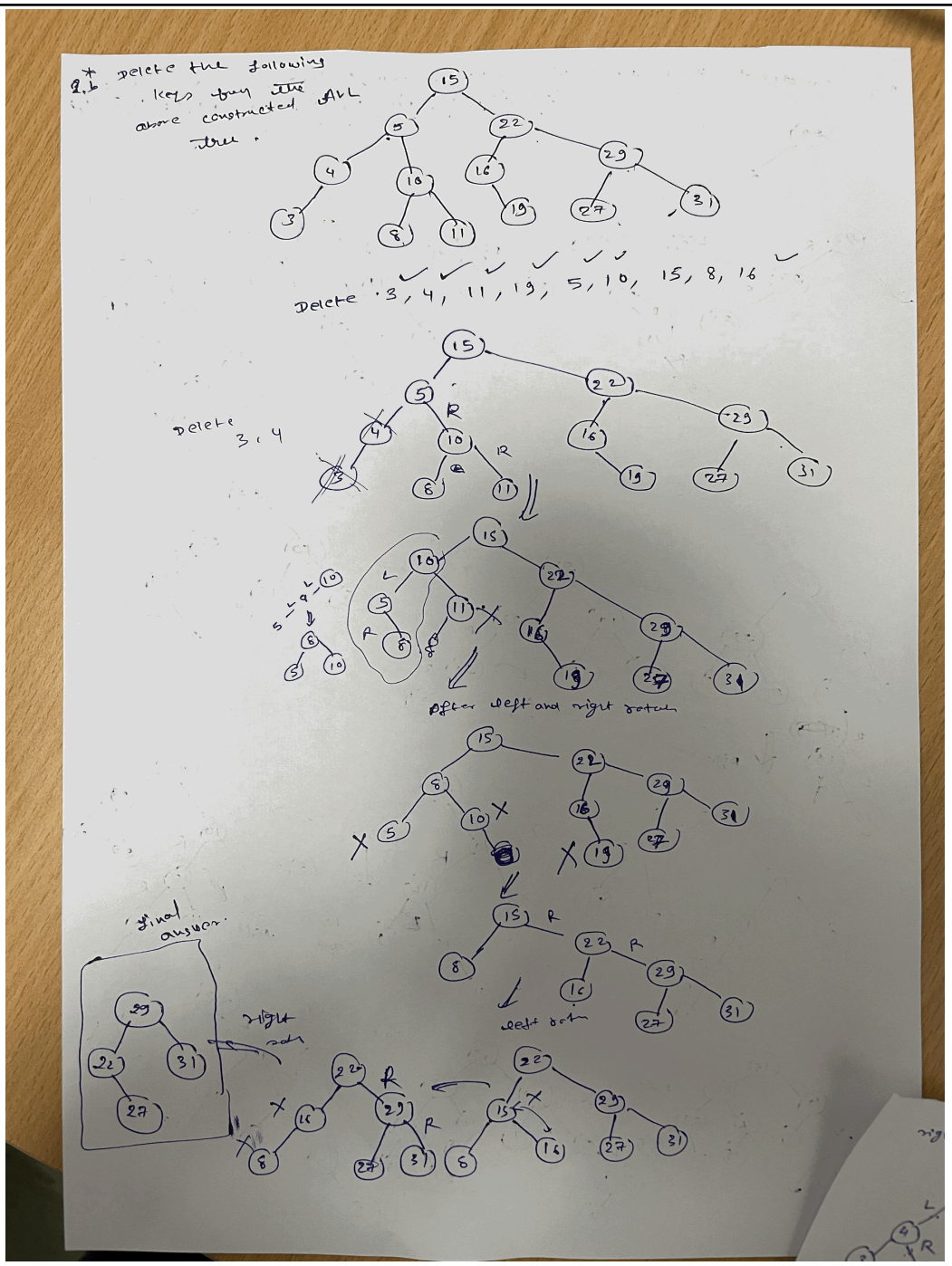
2	**a.** Construct AVL tree for the following keys 22, 27, 31, 10, 5, 15, 29, 19, 16, 11, 3, 4, 8	**[3 Marks]**

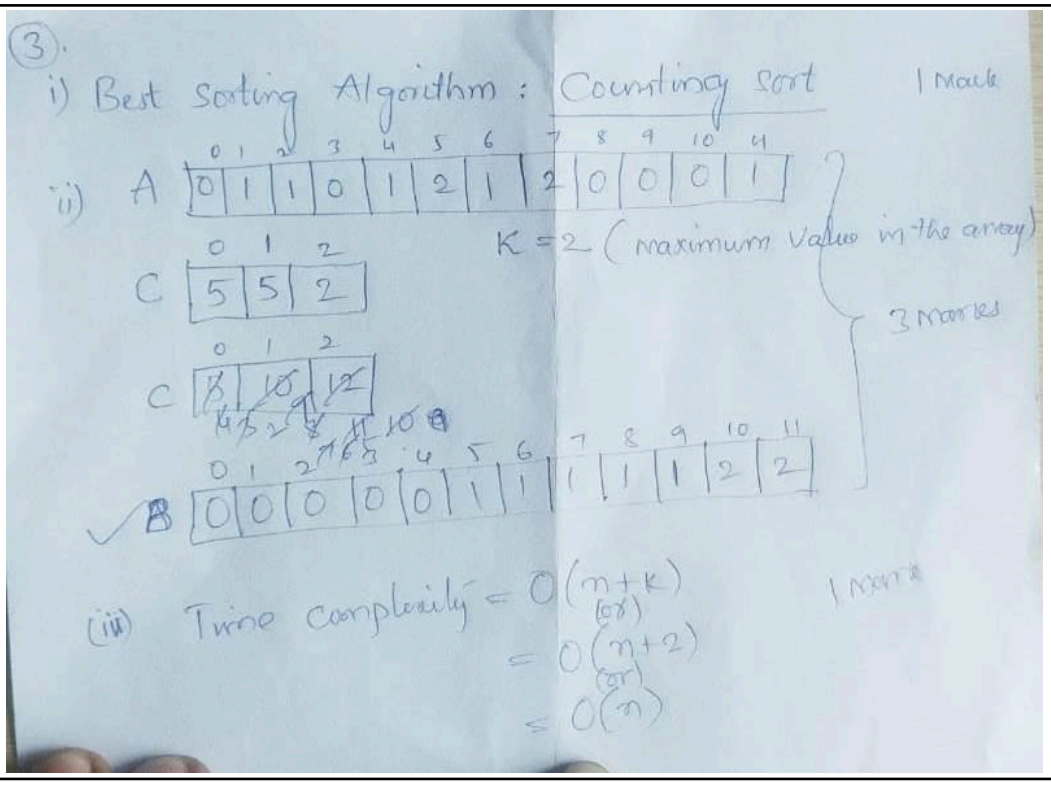

2.a Construct AVL tree for the following keys.
22, 27, 31, 10, 5, 15, 29, 19, 16, 11, 3, 4, 8
Left rotate
Right rotate
left rotate
right rotate
right rotate
left rotate
Right rot
left rotate
left rotate
left rotate
right rot
left
right rot



b. Delete the following keys from the above constructed AVL tree
3, 4, 11, 19, 5, 10, 15, 8, 16

[2 Marks]

		
3	Given an array A[] consisting of 0's, 1's and 2's, give an algorithm for sorting A[]. The algorithm should put all 0's first, then all 1's and all 2's last. Suggest a best sorting algorithm and sort the array A = {0,1,1,0,1,2,1,2,0,0,0,1} step-by-step by using this algorithm. What is the time complexity of the suggested sorting algorithm?	**[5 Marks]**


3.
i) Best Sorting Algorithm : Counting Sort
1 Mark
ii) A
K = 2 (maximum value in the array)
3 marks
C
B
(iii) Time complexity = O(n+k)
(or)
= O(n+2)
(or)
= O(n)
1 Mark