

# DATA STRUCTURE



## ADS | CCEE Practice Test - IV

Total points 13/20 ?

Duration: 30 Mins

The respondent's email (chitransh1709@gmail.com) was recorded on submission of this form.

0 of 0 points

Centre \*

Kharghar

PRN \*

240840320030

Name \*

Chitransh Mrigank Singh

### Questions

13 of 20 points

✗ Consider an AVL tree that needs to maintain its balanced property while inserting the following elements in the specified order: 38, 53, 43, 28, 33, 63, 81, 23, 31. After performing all the insertions, how many rotations would be required to ensure the AVL tree remains balanced? \*0/1

☒ 2 left rotations, 2 right rotations ✗

☐ 2 left rotations, 3 right rotations

☐ 3 left rotations, 2 right rotations

☐ 3 left rotations, 1 right rotation

Correct answer

☒ 3 left rotations, 2 right rotations

✗ In a binary min-heap with 103 unique elements, let K represent the index in the array where the largest element is stored. How many possible values can K take in this scenario? \*0/1

☐ 53

☐ 52

☐ 27

☒ 1 ✗

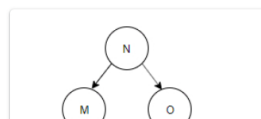
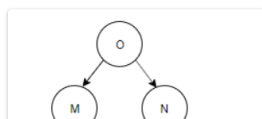
Correct answer

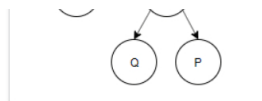
☒ 52

✓ Construct a binary tree by using postorder and inorder sequences given below. \*1/1

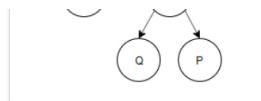
Inorder: N, M, P, O, Q

Postorder: N, P, Q, O, M

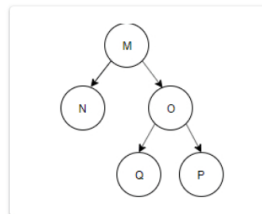




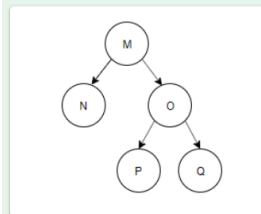
☐ Option 1



☐ Option 2



☐ Option 3



☒ Option 4 ✓

✗ What is the total number of distinct binary trees that can be constructed using four unlabelled nodes? \*0/1

☒ 10 ✗

☐ 14

☐ 13

☐ 12

Correct answer

☒ 14

✓ The height of a binary tree is the maximum number of edges in any root to leaf path. The maximum number of nodes in a binary tree of height h is: \*1/1

☐  $2^h - 1$

☐  $2^{(h-1)} - 1$

☒  $2^{(h+1)} - 1$  ✓

☐  $2^{(h+1)}$

✓ What will be the output when `chinTapakDum(new int[]{4, 1, 2, 1, 2})` is called? \*1/1

```
int chinTapakDum(int[] arr) {
```

```
    int result = 0;
```

```
    for (int num : arr) {
```

```
        result ^= num;
```

```
    }
```

```
    return result;
```

```
}
```

```
int finalDum = chinTapakDum(new int[]{4, 1, 2, 1, 2});
```

```
System.out.println(finalDum);
```

☒ 4 ✓

☐ 1

☐ 2

☐ 3

✗ What will be the result of the following operation? \*  
`Top(Push(T, Y))`

0/1

☐ Y

⌵

☐ Y + T

☒ T

✖

☐ YT

Correct answer

☒ Y

✓ Consider the Binary Search algorithm, which is designed to operate on sorted arrays. If you were to evaluate its performance in terms of efficiency: \*1/1

**For a scenario where the element is not found or is located at the last position**, think about how many comparisons would be required relative to the number of elements in the array.

**In a typical case where the target element is somewhere in the middle of the search process**, reflect on the expected number of comparisons needed.

Based on your analysis, what can be inferred about the time complexity of the Binary Search algorithm in terms of both worst-case and average-case scenarios?

☐  $O(n^2)$

☐  $O(1)$

☐  $O(n \log n)$

☒  $O(\log n)$

✓

✖ The preorder traversal of a binary search tree is 15, 10, 12, 11, 20, 18, 16, 19. Which one of the following is the postorder traversal of the tree? \*0/1

☐ 20, 19, 18, 16, 15, 12, 11, 10

☐ 10, 11, 12, 15, 16, 18, 19, 20

☐ 11, 12, 10, 16, 19, 18, 20, 15

☒ 19, 16, 18, 20, 11, 12, 10, 15

✖

Correct answer

☒ 11, 12, 10, 16, 19, 18, 20, 15

✓ If you were tasked with determining the total number of nodes  $N$  in a full binary tree, given that there are  $L$  leaves, which of the following equations would best describe this relationship? \*1/1

☐  $N = 2 * L$

☐  $N = L + 1$

☐  $N = L - 1$

☒  $N = 2 * L - 1$

✓

✓ What is the worst case time complexity of inserting a node in a doubly linked list? \*1/1

☐  $O(n \log n)$

☐  $O(\log n)$

☒  $O(n)$

✓

☐  $O(1)$

✓ class MyStack { \* 1/1

```
protected static final int MAX_SIZE = 150;

protected int count, index = -1;

protected Object elements[];
```

```

public MyStack() {
    elements = new Object[MAX_SIZE];
}

public void add(Object item) {
    if (count == MAX_SIZE) {
        System.out.println("Stack overflow");
        return;
    } else {
        index++;
        elements[index] = item;
        count++;
    }
}

public Object remove() {
    if (index < 0) {
        return null;
    } else {
        Object item = elements[index];
        index--;
        count--;
        return item;
    }
}

public class StackTest {
    public static void main(String args[]) {
        MyStack myStack = new MyStack();
        myStack.add("First");
        myStack.add("Second");
        Object element1 = myStack.remove();
        Object element2 = myStack.remove();
        Object element3 = myStack.remove();
        System.out.println(element3);
    }
}

```

**What will be the output of the StackTest class?**

- ☐ Second
- ☐ First
- ☒ null
- ☐ Stack overflow



✓ What will be the output when `aeHelloPadhlo(new int[]{3, 7, 1, 2, 8, 4, 5})` is called? \*1/1

```

int aeHelloPadhlo(int[] arr) {

```

```

int n = arr.length + 1;

int expectedSum = (n * (n + 1)) / 2;

int actualSum = 0;

for (int num : arr) {

    actualSum += num;

}

return expectedSum - actualSum;

}

int padhneKeBaad = aeeHelloPadhlo(new int[]{3, 7, 1, 2, 8, 4, 5});

System.out.println(padhneKeBaad);

```

- ☒ 6 ✓
- ☐ 9
- ☐ 4
- ☐ 5

✓ Which of the following insertion sequences will **not** require any rotations \*1/1 to maintain balance when inserting the elements {3, 4, 5, 6, 7, 8, 9} into an empty AVL tree?

- ☒ 6, 4, 8, 3, 5, 7, 9 ✓
- ☐ 6, 3, 5, 4, 9, 7, 8
- ☐ 9, 8, 7, 6, 5, 4, 3
- ☐ 3, 4, 5, 6, 7, 8, 9

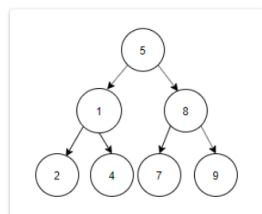
✗ In a full binary tree, If you were to derive a formula to express the number \*0/1 of leaves in relation to the number of internal nodes, which of the following relationships would accurately represent this connection?

- ☐  $L = 2 * I$
- ☐  $L = I + 1$
- ☐  $L = I - 1$
- ☒  $L = 2 * I - 1$  ✗

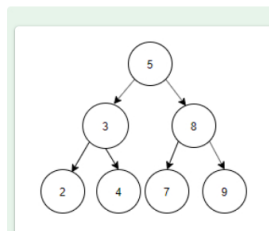
Correct answer

- ☒  $L = I + 1$

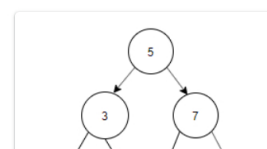
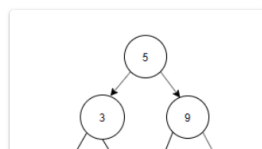
✓ Construct a binary search tree by using postorder sequence given below. \* 1/1 Postorder: 2, 4, 3, 7, 9, 8, 5.



☐ Option 1

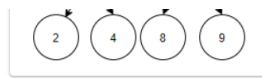


☒ Option 2 ✓





☐ Option 3



☐ Option 4

✓ You are given an unsorted array containing  $n$  distinct integers. You need to determine the maximum value in the array using a single traversal of the elements. Which of the following option accurately describes the time complexity of this operation? \*1/1

☐  $O(1)$

☐  $O(\log n)$

☒  $O(n)$  ✓

☐  $O(n \log n)$

✗ Which one of the following sequences, when stored in an array at locations  $A[1]$ ,  $A[2]$ ,  $A[3]$ ...,  $A[10]$ , forms a max-heap? \*0/1

☐ 28, 22, 19, 12, 18, 15, 6, 10, 11, 17

☐ 28, 22, 19, 10, 18, 15, 6, 11, 12, 17

☒ 28, 19, 22, 12, 18, 15, 6, 10, 11, 17 ✗

☐ 22, 28, 19, 12, 18, 15, 10, 11, 6, 17

Correct answer

☒ 28, 22, 19, 12, 18, 15, 6, 10, 11, 17

✓ What is the best-case time complexity of the Linear search? \* 1/1

☐  $O(n)$

☒  $O(1)$  ✓

☐  $O(n \log n)$

☐  $O(n^2)$

✓ The Binary Search algorithm is employed to find an element in a sorted array efficiently. What type of approach does it utilize to achieve this? \*1/1

☐ Linear way to search elements

☒ Divide and Conquer way to search elements ✓

☐ Sort and search Linearly

☐ Greedy search algorithm

☐ None of the above

Feedback of Mock

0 of 0 points

How was your Mock's experience? (No one word answer) \*

Need Revision

Level of exam \*

☐ Easy

☒ Moderate

☐ Tough

I understand the responsibility towards my life & everyone around me. I promise, I \*

am sincere towards my studies.

☒ Yes

☐ Other: .....

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

Google Forms

