PROBLEMS OF THE DAY -1

1. Largest Pair Sum

Find the largest pair sum in an array of distinct integers.

Examples :

Input: arr[] = [12, 34, 10, 6, 40]

Output: 74

Explanation: Sum of 34 and 40 is the largest, i.e, 34 + 40 = 74.

Input: arr[] = [10, 20, 30]

Output: 50

Explanation: 20 + 30 = 50.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**

2.Total count

You are given an array arr[] of positive integers and a threshold value k. For each element in the array, divide it into the minimum number of small integers such that each divided integer is less than or equal to k. Compute the total number of these integer across all elements of the array.

Examples:

Input: k = 3, arr[] = [5, 8, 10, 13]

Output: 14

Explanation: Each number can be expressed as sum of different numbers less than or equal to k as 5 (3 + 2), 8 (3 + 3 + 2), 10 (3 + 3 + 3 + 1), 13 (3 + 3 + 3 + 3 + 1). So, the sum of count of each element is (2+3+4+5)=14.

Input: k = 4, arr[] = [10, 2, 3, 4, 7]

Output: 8

Explanation: Each number can be expressed as sum of different numbers less than or equal to k as 10 (4 + 4 + 2), 2 (2), 3 (3), 4 (4) and 7 (4 + 3).So, the sum of count of each element is (3 + 1 + 1 + 1 + 2) = 8.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**

3.Roof Top

You are given the heights of consecutive buildings. You can move from the roof of a building to the roof of the next adjacent building. You need to find the maximum number of consecutive steps you can put forward such that you gain an increase in altitude with each step.

Examples:

Input: arr[] = [1, 2, 2, 3, 2]

Output: 1

Explanation: 1, 2, or 2, 3 are the only consecutive buildings with increasing heights thus maximum number of consecutive steps with an increase in gain in altitude would be 1 in both cases.

Input: arr[] = [1, 2, 3, 4]

Output: 3

Explanation: 1 to 2 to 3 to 4 is the jump of length 3 to have a maximum number of buildings with increasing heights, so maximum steps with increasing altitude becomes 3.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**

4.Facing The Sun

Given an array height representing the heights of buildings. You have to count the buildings that will see the sunrise (Assume the sun rises on the side of the array starting point).

Note: The height of the building should be strictly greater than the height of the buildings left in order to see the sun.

Examples :

Input: height = [7, 4, 8, 2, 9]

Output: 3

Explanation: As 7 is the first element, it can see the sunrise. 4 can't see the sunrise as 7 is hiding it. 8 can see. 2 can't see the sunrise. 9 also can see

the sunrise.

Input: height = [2, 3, 4, 5]

Output: 4

Explanation: As 2 is the first element, it can see the sunrise. 3 can see the sunrise as 2 is not hiding it. Same for 4 and 5, they also can see the sunrise.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**

5.Reverse Words

Given a String str, reverse the string without reversing its individual words. Words are separated by dots.

Note: The last character has not been '.'.

Examples :

Input: str = i.like.this.program.very.much

Output: much.very.program.this.like.i

Explanation: After reversing the whole string(not individual words), the input string becomes much.very.program.this.like.i

Input: str = pqr.mno

Output: mno.pqr

Explanation: After reversing the whole string , the input string becomes mno.pqr

Expected Time Complexity: **O(|str|)**

Expected Auxiliary Space: **O(|str|)**

6.Parenthesis Checker

Given an expression string x. Examine whether the pairs and the orders of {,},(,),[,] are correct in exp.

Examples :

Input: {([])}

Output: true

Explanation: { ([ ]) }. Same colored brackets can form balanced pairs, with 0 number of unbalanced bracket.

Input: ()

Output: true

Explanation: (). Same bracket can form balanced pairs,and here only 1 type of bracket is present and in balanced way.

Input: ([]

Output: false

Explanation: ([]. Here square bracket is balanced but the small bracket is not balanced and Hence , the output will be unbalanced.

Expected Time Complexity: **O(|x|)**

Expected Auixilliary Space: **O(|x|)**

7.Alternate positive and negative numbers

Given an unsorted array arr containing both positive and negative numbers. Your task is to rearrange the array and convert it into an array of alternate positive and negative numbers without changing the relative order.

Note:

- Resulting array should start with a positive integer (0 will also be considered as a positive integer).

- If any of the positive or negative integers are exhausted, then add the remaining integers in the answer as it is by maintaining the relative order.

- The array may or may not have equal number of positive and negative integers.

Examples:

Input: arr[] = [9, 4, -2, -1, 5, 0, -5, -3, 2]

Output: 9 -2 4 -1 5 -5 0 -3 2

Explanation: The positive numbers are [9, 4, 5, 0, 2] and the negative integers are [-2, -1, -5, -3]. Since, we need to start with the positive integer first

and then negative integer and so on (by maintaining the relative order as well), hence we will take 9 from the positive set of elements and then

-2 after that 4 and then -1 and so on.

Hence, the output is 9, -2, 4, -1, 5, -5, 0, -3, 2.

Input: arr[] = [-5, -2, 5, 2, 4, 7, 1, 8, 0, -8]

Output: 5 -5 2 -2 4 -8 7 1 8 0

Explanation : The positive numbers are [5, 2, 4, 7, 1, 8, 0] and the negative integers are [-5,-2,-8]. According to the given conditions we will start

from the positive integer 5 and then -5 and so on. After reaching -8 there are no negative elements left, so according to the given rule, we will

add the remaining elements (in this case positive elements are remaining) as it in by maintaining the relative order.

Hence, the output is 5, -5, 2, -2, 4, -8, 7, 1, 8, 0.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(n)**

8.Mirror Tree

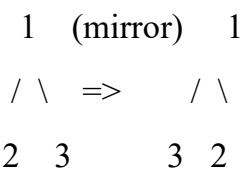
Given a Binary Tree, convert it into its mirror.

Examples:

Input:

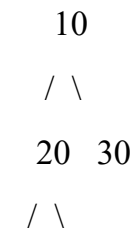
Output: 3 1 2

Explanation: The tree is



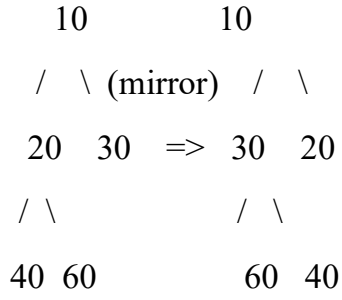
The inorder of mirror is 3 1 2

Input:



Output: 30 10 60 20 40

Explanation: The tree is



The inroder traversal of mirror is: 30 10 60 20 40.

Expected Time Complexity: **O(n)**

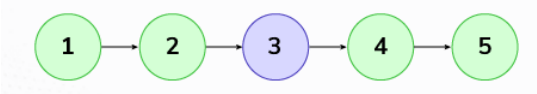
Expected Auxiliary Space: **O(height of the tree)**

9.Middle of a Linked List

Given the head of a linked list, the task is to find the middle. For example, the middle of 1-> 2->3->4->5 is 3. If there are two middle nodes (even count), return the second middle. For example, middle of 1->2->3->4->5->6 is 4.

Examples:

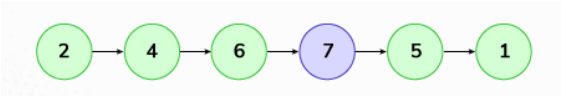
Input: Linked list: 1->2->3->4->5



Output: 3

Explanation: The given linked list is 1->2->3->4->5 and its middle is 3.

Input: Linked list: 2->4->6->7->5->1



Output: 7

Explanation: The given linked list is 2->4->6->7->5->1 and its middle is 7.

Expected Time Complexity: **O(n)**

Expected Auxiliary Space: **O(1)**

10.Minimum Cost Of Ropes

Given an array arr containing the lengths of the different ropes, we need to connect these ropes to form one rope. The cost to connect two ropes is equal to sum of their lengths. The task is to connect the ropes with minimum cost.

Examples:

Input: arr[] = [4, 3, 2, 6]

Output: 29

Explanation: We can connect the ropes in following ways.

1) First connect ropes of lengths 2 and 3. Which makes the array [4, 5, 6]. Cost of this operation 2 + 3 = 5.

2) Now connect ropes of lengths 4 and 5. Which makes the array [9, 6]. Cost of this operation 4 + 5 = 9.

3) Finally connect the two ropes and all ropes have connected. Cost of this operation 9 + 6 =15

Total cost for connecting all ropes is 5 + 9 + 15 = 29. This is the optimized cost for connecting ropes.

Other ways of connecting ropes would always have same or more cost. For example, if we connect 4 and 6 first (we get three rope of 3, 2 and 10), then connect 10 and 3 (we gettwo rope of 13 and 2). Finally we connect 13 and 2. Total cost in this way is 10 + 13 + 15 = 38.

Input: arr[] = [4, 2, 7, 6, 9]

Output: 62

Explanation: First, connect ropes 4 and 2, which makes the array [6, 7, 6, 9]. Cost of this operation 4 + 2 = 6.

Next, add ropes 6 and 6, which results in [12, 7, 9]. Cost of this operation 6 + 6 = 12.

Then, add 7 and 9, which makes the array [12,16]. Cost of this operation 7 + 9 = 16. And

finally, add these two which gives [28]. Hence, the total cost is 6 + 12 + 16 + 28 = 62.

Expected Time Complexity: **O(n logn)**

Expected Auxilliary Space: **O(n)**