1.Container With Most Water You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water.

Sol:- def max\_area(height):

max\_area = 0

left = 0

right = len(height) - 1

while left < right:

width = right - left

h = min(height[left], height[right])

max\_area = max(max\_area, width \* h)

if height[left] < height[right]:

left += 1

else:

right -= 1

return max\_area

height = [1, 8, 6, 2, 5, 4, 8, 3, 7]

print(max\_area(height))

2. Integer to Roman Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Sol:-

class Solution:

def intToRoman(self, num: int) -> str:

val = [

1000, 900, 500, 400,

100, 90, 50, 40,

10, 9, 5, 4, 1

]

syms = [

"M", "CM", "D", "CD",

"C", "XC", "L", "XL",

"X", "IX", "V", "IV", "I"

]

roman\_num = ''

i = 0

while num > 0:

for \_ in range(num // val[i]):

roman\_num += syms[i]

num -= val[i]

i += 1

return roman\_num

3. Roman to Integer Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M

Sol:-

def roman\_to\_int(s):

roman\_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}

result = 0

prev\_value = 0

for char in s:

value = roman\_dict[char]

result += value

if value > prev\_value:

result -= 2 \* prev\_value

prev\_value = value

return result

print(roman\_to\_int("III")) # Output: 3

print(roman\_to\_int("IV")) # Output: 4

print(roman\_to\_int("IX")) # Output: 9

print(roman\_to\_int("LVIII")) # Output: 58

print(roman\_to\_int("MCMXCIV")) # Output: 1994

4. Longest Common Prefix Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "". Example 1: Input: strs = ["flower","flow","flight"] Output: "fl"

Sol:-

def longestCommonPrefix(strs):

if not strs:

return ""

strs.sort()

prefix = ""

for i in range(len(strs[0])):

if strs[0][i] == strs[-1][i]:

prefix += strs[0][i]

else:

break

return prefix

strs = ["flower", "flow", "flight"]

output = longestCommonPrefix(strs)

print(output)

5. 3Sum Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0. Notice that the solution set must not contain duplicate triplets.

def threeSum(nums):

nums.sort()

res = []

for i in range(len(nums)-2):

if i > 0 and nums[i] == nums[i-1]:

continue

l, r = i+1, len(nums)-1

while l < r:

total = nums[i] + nums[l] + nums[r]

if total < 0:

l += 1

elif total > 0:

r -= 1

else:

res.append([nums[i], nums[l], nums[r]])

while l < r and nums[l] == nums[l+1]:

l += 1

while l < r and nums[r] == nums[r-1]:

r -= 1

l += 1

r -= 1

return res

nums = [-1, 0, 1, 2, -1, -4]

print(threeSum(nums))

6. . 3Sum Closest Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target. Return the sum of the three integers. You may assume that each input would have exactly one solution.

def threeSumClosest(nums, target):

nums.sort()

closest\_sum = float('inf')

for i in range(len(nums) - 2):

left, right = i + 1, len(nums) - 1

while left < right:

current\_sum = nums[i] + nums[left] + nums[right]

if abs(target - current\_sum) < abs(target - closest\_sum):

closest\_sum = current\_sum

if current\_sum < target:

left += 1

else:

right -= 1

return closest\_sum

nums = [-1, 2, 1, -4]

target = 1

print(threeSumClosest(nums, target))

7. Letter Combinations of a Phone Number Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order. A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.

from itertools import product

def letterCombinations(digits):

if not digits:

return []

phone = {'2': ['a', 'b', 'c'],

'3': ['d', 'e', 'f'],

'4': ['g', 'h', 'i'],

'5': ['j', 'k', 'l'],

'6': ['m', 'n', 'o'],

'7': ['p', 'q', 'r', 's'],

'8': ['t', 'u', 'v'],

'9': ['w', 'x', 'y', 'z']}

return [''.join(p) for p in product(\*(phone[d] for d in digits))]

digits = "23"

print(letterCombinations(digits))

8. Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: ● 0 <= a, b, c, d < n ● a, b, c, and d are distinct. ● nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order

def fourSum(nums, target):

nums.sort()

result = []

n = len(nums)

for i in range(n-3):

if i > 0 and nums[i] == nums[i-1]:

continue

for j in range(i+1, n-2):

if j > i+1 and nums[j] == nums[j-1]:

continue

left = j + 1

right = n - 1

while left < right:

total = nums[i] + nums[j] + nums[left] + nums[right]

if total == target:

result.append([nums[i], nums[j], nums[left], nums[right]])

while left < right and nums[left] == nums[left+1]:

left += 1

while left < right and nums[right] == nums[right-1]:

right -= 1

left += 1

right -= 1

elif total < target:

left += 1

else:

right -= 1

return result

nums = [1, 0, -1, 0, -2, 2]

target = 0

print(fourSum(nums, target))

9. Remove Nth Node From End of List Given the head of a linked list, remove the nth node from the end of the list and return its head

class ListNode:

def \_\_init\_\_(self, val=0, next=None):

self.val = val

self.next = next

def removeNthFromEnd(head, n):

dummy = ListNode(0)

dummy.next = head

first = dummy

second = dummy

for i in range(1, n + 2):

while first is not None:

first = first.next

second = second.next

second.next = second.next.next

return dummy.next

10. Valid Parentheses Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if: 1. Open brackets must be closed by the same type of brackets. 2. Open brackets must be closed in the correct order. 3. Every close bracket has a corresponding open bracket of the same type. Example 1: Input: s = "()" Output: true

def isValid(s):

stack = []

mapping = {")": "(", "}": "{", "]": "["}

for char in s:

if char in mapping:

top\_element = stack.pop() if stack else '#'

if mapping[char] != top\_element:

return False

else:

stack.append(char)

return not stack

s = "()"

print(isValid(s)) # Output: True