**1. 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.**

**Return *the maximum amount of water a container can store*.**

**Notice that you may not slant the container.**

**Program:**

**def maxArea(height):**

**left, right = 0, len(height) - 1**

**max\_water = 0**

**while left < right:**

**width = right - left**

**max\_water = max(max\_water, min(height[left], height[right]) \* width)**

**if height[left] < height[right]:**

**left += 1**

**else:**

**right -= 1**

**return max\_water**

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

**x=maxArea(height)**

**print(x)**

**Output:**

1. **Integer to Roman**

**Program:**

**def roman(num):**

**val = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]**

**symb = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"]**

**roman\_num = ""**

**for i in range(len(val)):**

**while num >= val[i]:**

**num -= val[i]**

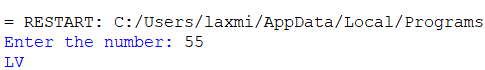
**roman\_num += symb[i]**

**return roman\_num**

**num=int(input("Enter the number: "))**

**print(roman(num))**

**Output:**

****

1. **Roman to Integer**

**Program:**

**def inte(s):**

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

**total = 0**

**for i in range(len(s)):**

**if i > 0 and roman[s[i]] > roman[s[i - 1]]:**

**total += roman[s[i]] - 2 \* roman[s[i - 1]]**

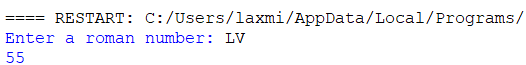
**else:**

**total += roman[s[i]]**

**return total**

**roman=input("Enter a roman number: ")**

**print(inte(roman))**

**Output:**

1. **Longest Common Prefix**

**Program:**

**def long(strs):**

**if not strs:**

**return ""**

**shortest = min(strs, key=len)**

**for i, ch in enumerate(shortest):**

**for other in strs:**

**if other[i] != ch:**

**return shortest[:i]**

**return shortest**

**n=int(input("Enter the range: "))**

**s=[]**

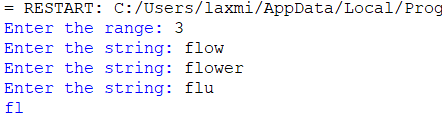
**for i in range(n):**

**x=input("Enter the string: ")**

**s.append(x)**

**print(long(s))**

**Output:**

****

1. **3Sum**

**Program:**

**def tsum(nums):**

**nums.sort()**

**res = []**

**for i in range(len(nums)):**

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

**continue**

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

**while left < right:**

**sum = nums[i] + nums[left] + nums[right]**

**if sum < 0:**

**left += 1**

**elif sum > 0:**

**right -= 1**

**else:**

**res.append([nums[i], 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**

**return res**

**n=int(input("Enter the range: "))**

**num=[]**

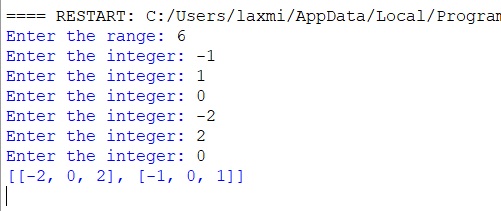
**for i in range(n):**

**x=int(input("Enter the integer: "))**

**num.append(x)**

**print(tsum(num))**

**Output:**

****

1. **3Sum Closest**

**Program:**

**def closest(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:**

**curr\_sum = nums[i] + nums[left] + nums[right]**

**if abs(curr\_sum - target) < abs(closest\_sum - target):**

**closest\_sum = curr\_sum**

**if curr\_sum < target:**

**left += 1**

**elif curr\_sum > target:**

**right -= 1**

**else:**

**return curr\_sum**

**return closest\_sum**

**n=int(input("Enter the range: "))**

**nums=[]**

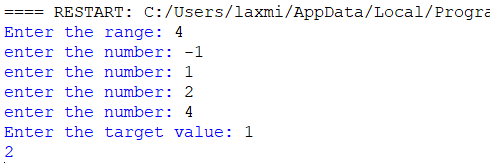
**for i in range(n):**

**x=int(input("enter the number: "))**

**nums.append(x)**

**target=int(input("Enter the target value: "))**

**print(closest(nums,target))**

**Output:**

1. **Letter Combinations of a Phone Number**

**Program:**

**def comb(digits):**

**if not digits:**

**return []**

**phone = {'2': 'abc', '3': 'def', '4': 'ghi', '5': 'jkl','6': 'mno', '7': 'pqrs', '8': 'tuv', '9': 'wxyz'}**

**result = ['']**

**for digit in digits:**

**result = [prefix + letter for prefix in result for letter in phone[digit]]**

**return result**

**digits=input("Enter the phone digits: ")**

**print(comb(digits))**

**Output:**

1. **4Sum**

**Program:**

**def fsum(nums, target):**

**nums.sort()**

**res, quad = [], []**

**def kSum(k, start, target):**

**if k != 2:**

**for i in range(start, len(nums) - k + 1):**

**if i > start and nums[i] == nums[i - 1]:**

**continue**

**quad.append(nums[i])**

**kSum(k - 1, i + 1, target - nums[i])**

**quad.pop()**

**return**

**left, right = start, len(nums) - 1**

**while left < right:**

**curr\_sum = nums[left] + nums[right]**

**if curr\_sum < target:**

**left += 1**

**elif curr\_sum > target:**

**right -= 1**

**else:**

**res.append(quad + [nums[left], nums[right]])**

**left += 1**

**right -= 1**

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

**left += 1**

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

**right -= 1**

**kSum(4, 0, target)**

**return res**

**n=int(input("Enter the range: "))**

**num=[]**

**for i in range(n):**

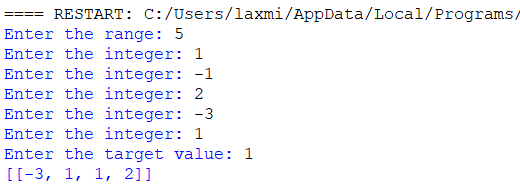
**x=int(input("Enter the integer: "))**

**num.append(x)**

**target=int(input("Enter the target value: "))**

**print(fsum(num,target))**

**Output:**

****

1. **Remove Nth Node From End of List**

**Program:**

**class ListNode:**

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

**self.val = val**

**self.next = next**

**def removeNthFromEnd(head, n):**

**dummy = ListNode(0, head)**

**first = second = dummy**

**for \_ in range(n + 1):**

**first = first.next**

**wturn dummy.next**

**def print\_linked\_list(head):**

**current = head**

**result = []**

**while current:**

**result.append(current.val)**

**current = current.next**

**print(result)**

**values = [1, 2, 3, 4, 5]**

**n = 2**

**head = create\_linked\_list(values)**

**print("Original list:")**

**print\_linked\_list(head)**

**new\_head = removeNthFromEnd(head, n)**

**print("Modified list after removing the nth node from the end:")**

**print\_linked\_list(new\_head)hile first:**

**first, second = first.next, second.next**

**second.next = second.next.next**

**return dummy.next**

**def create\_linked\_list(values):**

**dummy = ListNode(0)**

**current = dummy**

**for val in values:**

**current.next = ListNode(val)**

**current = current.next**

**return dummy.next**

**def print\_linked\_list(head):**

**current = head**

**result = []**

**while current:**

**result.append(current.val)**

**current = current.next**

**print(result)**

**values = [1, 2, 3, 4, 5]**

**n = 2**

**head = create\_linked\_list(values)**

**print("Original list:")**

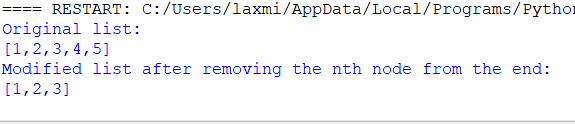
**print\_linked\_list(head)**

**new\_head = removeNthFromEnd(head, n)**

**print("Modified list after removing the nth node from the end:")**

**print\_linked\_list(new\_head)**

**Output:**

****

1. **Valid Parentheses**

**Program:**

**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:**

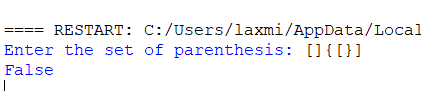
**stack.append(char)**

**return not stack**

**s=input("Enter the set of parenthesis: ")**

**print(isValid(s))**

**Output:**

****