**ASSIGNMENT 7**

Name: Aaryan Aher

PRN: 22210645

Div:- TY A [1]

Roll No.: 371003

**Title:** Find a subset of a given set S = {s1 ,s2 ,.....,s n } of n positive integers whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given Problem instance doesn't have a solution.

**CODE:**# Function to find all unique subsets of 'nums' that sum to 'target'

def find\_subset(nums, target):

    # Sort the input array to handle duplicates and to stop early when the current sum exceeds the target

    nums.sort()

    # List to store the current possible solution (subset)

    possible\_ans = []

    # Call the helper function to start the recursion

    helper(nums, target, 0, 0, possible\_ans)

# Recursive helper function to find subsets

def helper(nums, target, index, current\_sum, possible\_ans):

    # Base case: if current\_sum equals the target, we found a valid subset

    if current\_sum == target:

        print(possible\_ans)  # Print the current valid subset

        return

    # To avoid duplicates, we track the last element used

    prev\_element = -1

    # Iterate through the remaining elements starting from 'index'

    for i in range(index, len(nums)):

        # Skip duplicates: Only proceed if the current element is different from the previous one

        if prev\_element != nums[i]:

            # If adding nums[i] exceeds the target, no need to proceed further since the array is sorted

            if nums[i] + current\_sum > target:

                break

            # Add nums[i] to the current subset

            possible\_ans.append(nums[i])

            # Set the previous element to the current one to handle duplicates

            prev\_element = nums[i]

            # Recursively call helper with the updated sum and next index

            helper(nums, target, i + 1, current\_sum + nums[i], possible\_ans)

            # Backtrack by removing the last element (backtracking step)

            possible\_ans.pop()

# Driver code

if \_\_name\_\_ == "\_\_main\_\_":

    # Input list and the target sum

    nums = [1, 2, 5, 6, 8]

    # Call the function to find subsets that sum up to 9

    find\_subset(nums, 9)

**OUTPUT:**

