**6.8 Unique set of Numbers**

**Aim:** The aim is to find all unique sets of numbers from a given collection that add up to a specific target value, where each number from the collection can be used at most once.

**Algorithm:**

1. Let the input be an array of integers nums = [n₁, n₂, …, nk] and a target sum T.

2. Sort the array nums in non-decreasing order.

3. We define a recursive backtracking procedure: dfs(start, target, path, result)  
  - start → index in nums from which to consider numbers  
  - target → remaining sum we want to form  
  - path → current set of chosen numbers  
  - result → collection of all valid unique sets

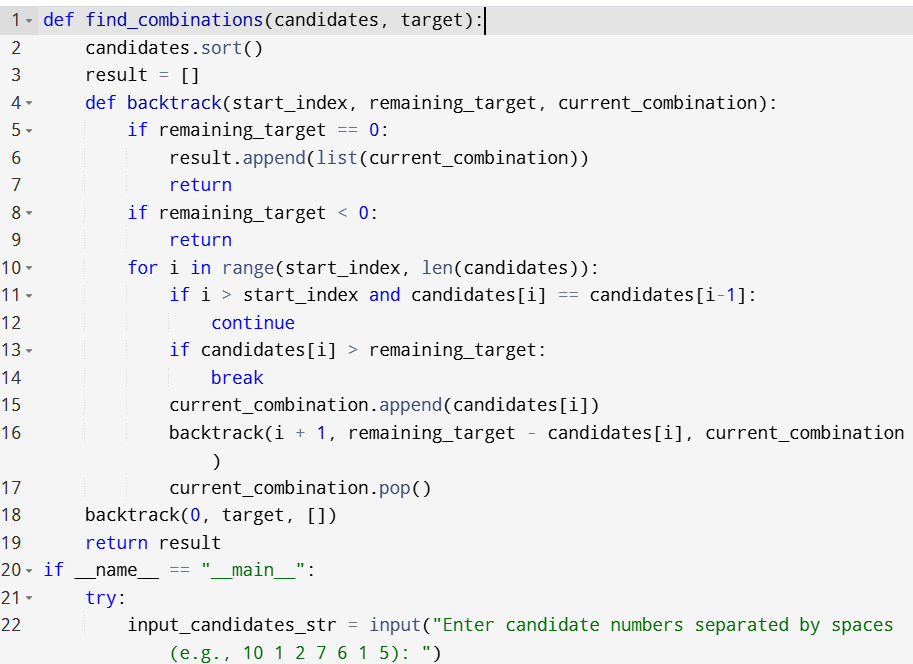
4. Base conditions:  
  - If target = 0: Add the current path to result (a valid set found). Return.  
  - If target < 0: Return immediately (no valid set).

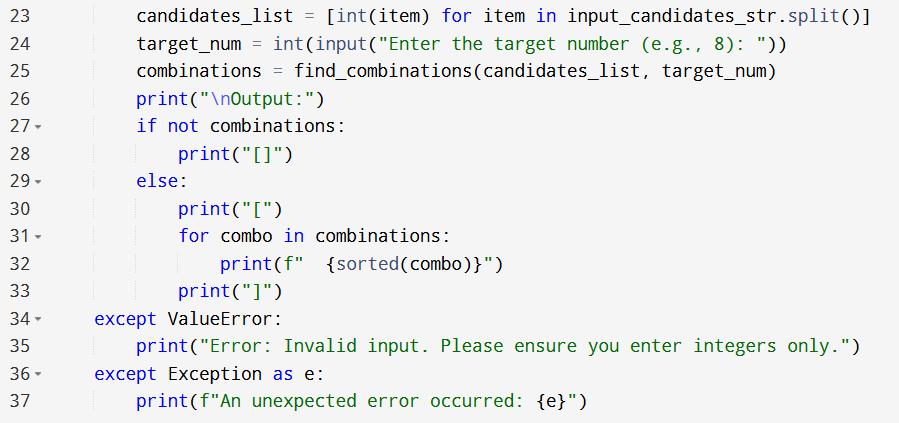
5. Recursive case:  
  For each index i from start to length of nums:  
  - If i > start and nums[i] = nums[i−1], skip this element to avoid duplicates.  
  - If nums[i] > target, break (further numbers will also be too large).  
  - Include nums[i] in the current path, and call  
    dfs(i+1, target − nums[i], path ∪ {nums[i]}, result)  
  - After recursion, remove nums[i] from path (backtrack).

6. Initially, call the procedure as: dfs(0, T, ∅, result)

7. The final answer is the collection result, containing all unique sets of numbers whose sum equals the target.

**Program:**

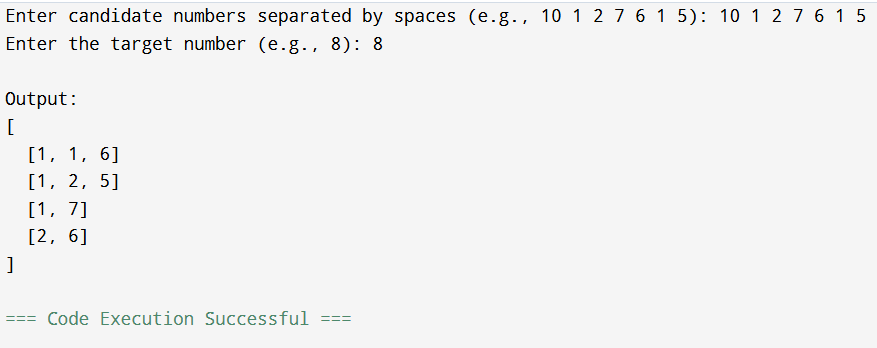
****



**Input:**

* Enter candidate numbers separated by spaces (e.g., 10 1 2 7 6 1 5):
* Enter the target number (e.g., 8):

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(n2^n)
* Space Complexity: O(n).