UNIVERSITY INSTITUTE of COMPUTING
Asia's Fastest Growing University

NAAC GRADE A+ ACCREDITED UNIVERSITY

Experiment No-5

Student Name: Raja Kumar **UID: 24MCA20229**
Branch: MCA **Section/Group: 24MCA-2(B)**
Semester: 2nd **Date Of Performance: 08/04/2025**
Subject Name: DAA LAb **Subject Code: 24CAP-612**

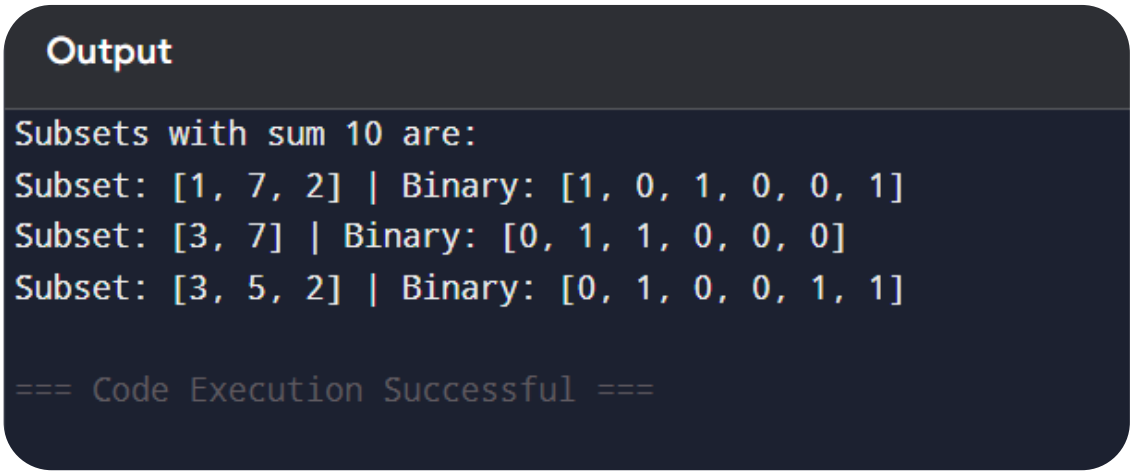
Aim: Find a subset of a given set S={sl,s2,....sn} of n positive integers whose sum is equal to a given positive integer d. For example, if S={1,2,5,6,8}and d =9 there are two solutions{1,2,6}and{1,8}.A suitable message is to be displayed if the given problem instance doesn't have a solution.

Input:

```
import java.util.*;

public class SubsetSumSolver {

    public static void findSubsets(int[] set, int target) {
        List<Integer> currentSubset = new ArrayList<>();
        List<Integer> binaryMask = new ArrayList<>(Collections.nCopies(set.length, 0));
        List<List<Integer>> allSolutions = new ArrayList<>();
        List<List<Integer>> allBinaryMasks = new ArrayList<>();

        findSubsetsHelper(set, 0, target, currentSubset, binaryMask, allSolutions,
allBinaryMasks);

        if (allSolutions.isEmpty()) {
            System.out.println("No subset found with the given sum.");
        } else {
            System.out.println("Subsets with sum " + target + " are:");
            for (int i = 0; i < allSolutions.size(); i++) {
                System.out.println("Subset: " + allSolutions.get(i) + " | Binary: " +
allBinaryMasks.get(i));
            }
        }
    }

    private static void findSubsetsHelper(int[] set, int index, int target,
                                          List<Integer> currentSubset, List<Integer> binaryMask,
                                          List<List<Integer>> allSolutions, List<List<Integer>>
allBinaryMasks) {

        if (target == 0) {
            allSolutions.add(new ArrayList<>(currentSubset));
            allBinaryMasks.add(new ArrayList<>(binaryMask));
            return;
        }
```

```
 if (index >= set.length || target < 0) {
 return;
 }

 // Include current element
 currentSubset.add(set[index]);
 binaryMask.set(index, 1);
 findSubsetsHelper(set, index + 1, target - set[index], currentSubset, binaryMask, allSolutions,
allBinaryMasks);

 // Backtrack and exclude current element
 currentSubset.remove(currentSubset.size() - 1);
 binaryMask.set(index, 0);
 findSubsetsHelper(set, index + 1, target, currentSubset, binaryMask, allSolutions,
allBinaryMasks);
 }

 public static void main(String[] args) {
 // Example input
 int[] set = {1,3,7,11,5,2};
 int targetSum = 10;

 findSubsets(set, targetSum);
 }
}
```

Output:



Leaning Outcomes:

- You learn how to use recursion and backtracking to find all subsets that match a given condition (sum).
- You understand how to represent subset choices using a binary mask (0 for excluded, 1 for included).
- You practice working with lists and dynamic data structures to store and display multiple solutions.