**Experiment 2.2**

**Aim:** *Develop a program and analyze complexity to implement subset-sum problem using Dynamic Programming.*

**Objectives:** *Objective is to implement subset-sum problem using Dynamic programming.*

**Input/Apparatus Used:** *VS CODE*

# Procedure/Algorithm:

# *So we will create a 2D array of size (arr.size() + 1) \* (target + 1) of type boolean. The state DP[i][j] will be true if there exists a subset of elements from A[0….i] with sum value = ‘j’. The approach for the problem is:*

# *if*

# *(A[i-1] > j) DP[i][j] = DP[i-1][j]*

# *else*

# *DP[i][j] = DP[i-1][j] OR DP[i-1][j-A[i-1]]*

# *This means that if current element has value greater than ‘current sum value’ we will copy the answer for previous cases*

# *And if the current sum value is greater than the ‘ith’ element we will see if any of previous states have already experienced the sum=’j’ OR any previous states experienced a value ‘j – A[i]’ which will solve our purpose.*

# Code:

# *#include <iostream>*

# *#include <vector>*

# *using namespace std;*

# *bool isSubsetSum(vector<int>& nums, int targetSum) {*

# *int n = nums.size();*

# *vector<vector<bool>> dp(n + 1, vector<bool>(targetSum + 1, false));*

# *// Initialize the DP table*

# *for (int i = 0; i <= n; i++)*

# *dp[i][0] = true;*

# *for (int i = 1; i <= n; i++) {*

# *for (int j = 1; j <= targetSum; j++) {*

# *// If the current number is greater than the target sum, skip it*

# *if (nums[i - 1] > j)*

# *dp[i][j] = dp[i - 1][j];*

# *else {*

# *// Include the current number in the sum or exclude it*

# *dp[i][j] = dp[i - 1][j] || dp[i - 1][j - nums[i - 1]];*

# *}*

# *}*

# *}*

# *// The final result is stored in dp[n][targetSum]*

# *return dp[n][targetSum];*

# *}*

# *int main() {*

# *vector<int> nums = {3, 34, 4, 12, 5, 2};*

# *int targetSum = 9;*

# *if (isSubsetSum(nums, targetSum))*

# *cout << "Subset with the given sum exists." << endl;*

# *else*

# *cout << "No subset with the given sum exists." << endl;*

# *return 0;*

# *}*

# Observations/Outcome :

# 

# Time Complexity:

* *Time Complexity: O(n \* targetSum),*
* *Space Complexity: O(n \* targetSum)*