**Subset Sum Problem**

**Medium**

Given an array of non-negative integers, and a value *sum*, determine if there is a subset of the given set with sum equal to given *sum*.

**Example 1:**

**Input**:

N = 6

arr[] = {3, 34, 4, 12, 5, 2}

sum = 9

**Output:** 1

**Explanation**: Here there exists a subset with

sum = 9, 4+3+2 = 9.

**Example 2:**

**Input**:

N = 6

arr[] = {3, 34, 4, 12, 5, 2}

sum = 30

**Output:** 0

**Explanation**: There is no subset with sum 30.

**Expected Time Complexity:** O(sum\*N)  
**Expected Auxiliary Space:** O(sum\*N)

**Constraints:**  
1 <= N <= 100  
1<= arr[i] <= 100  
1<= sum <= 105

**Company Tags**

[**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Microsoft**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Microsoft)

**Topic Tags**

[**Dynamic Programming**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Dynamic%20Programming) [**Algorithms**](https://practice.geeksforgeeks.org/explore/?category%5b%5d=Algorithms)

//{ Driver Code Starts

import java.io.\*;

import java.util.\*;

class CodingMaxima

{

public static void main(String args[])throws IOException

{

BufferedReader read = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(read.readLine());

while(t-- > 0)

{

int N = Integer.parseInt(read.readLine());

String input\_line[] = read.readLine().trim().split("\\s+");

int arr[]= new int[N];

for(int i = 0; i < N; i++)

arr[i] = Integer.parseInt(input\_line[i]);

int sum = Integer.parseInt(read.readLine());

Solution ob = new Solution();

if(ob.isSubsetSum(N, arr, sum))

System.out.println(1);

else

System.out.println(0);

}

}

}

// } Driver Code Ends

//User function Template for Java

class Solution{

static Boolean isSubsetSum(int n, int arr[], int sum){

boolean[][] dp = new boolean[n+1][sum+1];

Arrays.fill(dp[0], false);

dp[0][0] = true;

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

for (int j = 0; j <= sum; j++) {

if (arr[i-1] > j) {

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

} else {

dp[i][j] = dp[i-1][j] || dp[i-1][j-arr[i-1]];

}

}

}

return dp[n][sum];

}

}