**Problem link** : [Partition Equal Subset Sum](https://www.codingninjas.com/codestudio/problems/partition-equal-subset-sum_892980?source=youtube&campaign=striver_dp_videos&utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_dp_videos)

**Approach:** Acc to the ques we need to tell whether 2 subsets are such that both have sum = sum/2.

\*Sum = all elements

* If sum = odd, then 2 subsets can never be obtain such that their both sum will be equal to s/2.  **( 5/2 ??)**
* If sum = even, then only it can divided into 2 parts (s/2 + s/2) and we can check further if s/2 can be obtain by any subset and if one subset is there then rest of the element sum will be s/2 definitely.  **(6/2 =3)**

Sum = 10 (s)

One subset has sum = 5 (s/2)

Then definitely other subset can be formed with sum = 5 (s/2)

**S = s/2 + s/2**

**Now question has boiled down to DP14 just that here we have to find a subset such that its sum will be equal to s/2 (target)**

**bool subsetSumToK(int n, int k, vector<int> &arr) {**

**///prev will store the (ind-1) row to get values of [ind-1][target] and [ind-1][target-arr[ind]] = prev[k+1]**

**vector<bool> prev(k+1, 0), curr(k+1, 0);**

**prev[0] = curr[0] = 1; //target = 0, return 1;**

**prev[arr[0]] = 1; //ind=0, prev[arr[0]] = 1**

**//use nested for loops to build dp from bottom to up**

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

**for(int target= 1; target<=k; target++){**

**//explore**

**int notTake = prev[target];**

**int take = false;**

**if(arr[ind] <= target)**

**take = prev[target-arr[ind]];**

**curr[target] = (notTake || take);**

**}**

**//update prev**

**prev = curr;**

**}**

**return prev[k];**

**}**

**bool canPartition(vector<int> &arr, int n)**

**{**

**int sum = 0;**

**for(auto it: arr)**

**sum += it;**

**if(sum %2 == 1) return false;**

**int target = sum/2;**

**return subsetSumToK(n, target, arr);**

**}**