Combination Sum-I

Same as subset sum Decrement k and if it is less than 0 then it is a base case if k==0 it is the ans but if not we simply return.

```
#include <bits/stdc++.h>
void helper(int ind,vector<int> arr,int n,int k,vector<int> temp,vector<vector<int>>& ans)
{
    if(ind==n)
    {
        if(k==0)
        {
            ans.push_back(temp);
        }
        return;
    }
    temp.push_back(arr[ind]);
    helper(ind+1,arr,n,k-arr[ind],temp,ans);
    temp.pop_back();
    helper(ind+1,arr,n,k,temp,ans);
}
vector<vector<int>> findSubsetsThatSumToK(vector<int> arr, int n, int k)
{
    // Write your code here.
    vector<int> temp;
    vector<vector<int>> ans;
    helper(0,arr,n,k,temp,ans);
    return ans;
}
```

- **Time Complexity :** O(2^t*k) where t is the target, k is the average length

Reason: Assume if you were not allowed to pick a single element multiple times, every element will have a couple of options: pick or not pick which is 2^n different recursion calls, also assuming that the average length of every combination generated is k. (to put length k data structure into another data structure)

Why not (2^n) but (2^t) (where n is the size of an array)?

Assume that there is 1 and the target you want to reach is 10 so 10 times you can “pick or not pick” an element.

- **Space Complexity: O(k*x)**, k is the average length and x is the no. of combinations