





using System;  
using System.Collections.Generic;  
  
class Program  
{  
 static void Main()  
 {  
 int[] Candidates = { 10, 1, 2, 7, 6, 1, 5 };  
 int Target = 8;  
  
 List<List<int>> result = new List<List<int>>();  
 List<int> current = new List<int>();  
   
 Array.Sort(Candidates);  
 FindCombinations(Candidates, Target, 0, current, result);  
  
 foreach (List<int> combination in result)  
 {  
 Console.WriteLine(string.Join(", ", combination));  
 }  
 }  
  
 static void FindCombinations(int[] candidates, int target, int start, List<int> current, List<List<int>> result)  
 {  
 if (target == 0)  
 {  
 result.Add(new List<int>(current));  
 return;  
 }  
  
 for (int i = start; i < candidates.Length; i++)  
 {  
 if (candidates[i] > target)  
 {  
 break;  
 }  
  
 if (i > start && candidates[i] == candidates[i - 1])  
 {  
 continue;  
 }  
 current.Add(candidates[i]);  
 FindCombinations(candidates, target - candidates[i], i + 1, current, result);  
 current.RemoveAt(current.Count - 1);  
 }  
 }  
}

using System;  
using System.Collections.Generic;  
  
class Program  
{  
 static void Main()  
 {  
 int[] Candidates = { 2, 5, 2, 1, 2 };  
 int Target = 5;  
  
 List<List<int>> result = new List<List<int>>();  
 List<int> current = new List<int>();  
   
 Array.Sort(Candidates);  
 FindCombinations(Candidates, Target, 0, current, result);  
  
 foreach (List<int> combination in result)  
 {  
 Console.WriteLine(string.Join(", ", combination));  
 }  
 }  
  
 static void FindCombinations(int[] candidates, int target, int start, List<int> current, List<List<int>> result)  
 {  
 if (target == 0)  
 {  
 result.Add(new List<int>(current));  
 return;  
 }  
  
 for (int i = start; i < candidates.Length; i++)  
 {  
 if (candidates[i] > target)  
 {  
 break;  
 }  
  
 if (i > start && candidates[i] == candidates[i - 1])  
 {  
 continue;  
 }  
 current.Add(candidates[i]);  
 FindCombinations(candidates, target - candidates[i], i + 1, current, result);  
 current.RemoveAt(current.Count - 1);  
 }  
 }  
}

