Lab Assignment 3

1. Problem Description (knapsack problem, 背包问题)
   For a knapsack, you can put items in it with the maximum weight s. Now there exist n items, their weight is $w_1, w_2, \cdots, w_n$, respectively and assume $w_i(l, i, n)$ are all positive integers, which are put in a list w. You can define a recursive boolean function $knap(s_0, n_0)$, if the selected n_0 items are put into the knapsack and the total weight is exactly equal to s_0, the function returns true, and outputs a set of the weight for selected items. Otherwise, it returns false.
   Suppose s is 20 kg, n is 8 items, and the weight (kg) is a set of generated and non-repeated random positive integers between 1 and 25, such as 13, 8, 6, 2, 25, 19, 20, 1.
   Please design an algorithm to resolve this problem and then implement it by Python.

   Example:
   - Input:
     ```
     s = 20
     w = [13, 8, 6, 2, 25, 19, 20, 1]
     ```
     Output:
     ```
     There is a solution:
     20 = 19 + 1
     ```
   - Input:
     ```
     s = 20
     w = [24, 17, 23, 8, 11, 15, 16, 25]
     ```
     Output:
     ```
     There is no solution.
     ```

   Hint:
   This problem can be recursively defined as follows:

$$\text{knap}(s_0, n_0) = \begin{cases} \text{true}, & \text{if } s_0 = 0, \text{successful} \\ \text{false}, & \text{if } s_0 \le 0, \text{unsuccessful} \\ \text{false}, & \text{if } s_0 > 0 \text{ and } n_0 \le 0, \text{unsuccessful} \\ \text{knap}(s_0, n_0 - 1), & \text{exclusive of } w_n \text{ in selected items} \\ \text{knap}(s_0 - w_n, n_0 - 1), & \text{inclusive of } w_n \text{ in selected items} \end{cases}$$

   The first three lines in the above definition are boundary conditions (i.e. base case). The first line means that we don't put any item in the knapsack. The second line denotes that it is impossible that the total weight is less than zero. The third line indicates that if we don't put any item in the knapsack, the total weight cannot be greater than zero.

   Therefore, the definition is deterministic, because recursive procedure is executed each time: n_0 minus 1, s_0 minus w_n possibly. Until s_0 is less than 0 or n_0 is equal to 0, we can use the boundary conditions to determine the final value. Note that it is possible that the final output result about weight of each item is not unique.