Suggested Solutions to HW #4

3. (5.17) The Knapsack Problem that we discussed in class is defined as follows: Given a set S of n items, where the ith item has an integer size $S[i]$, and an integer K, find a subset of the items whose sizes sum to exactly K or determine that no such subset exists.

We have described in class an algorithm to solve the problem. Modify the algorithm to solve a variation of the knapsack problem where each item has an *unlimited* supply. In your algorithm, please change the type of $P[i,k].belong$ into integer and use it to record the number of copies of item i needed.
Solution.

Algorithm Knapsack (S, K)**;**

begin
  $P[0,0].exist := true$;
  $P[0,0].belong := 0$;
  for $k := 1$ **to** K **do**
    $P[0,k].exist := false$;
  for $i := 1$ **to** n **do**
    for $k := 0$ **to** K **do**
      $P[i,k].exist := false$;
      if $P[i-1,k].exist$ **then**
        $P[i,k].exist := true$;
        $P[i,k].belong := 0$;
      else if $k - S[i] \geq 0$ **then**
        if $P[i, k-S[i]].exist$ **then**
          $P[i,k].exist := true$;
          $P[i,k].belong := P[i,k-S[i]].belong + 1$;
end

□

4. (5.20) Let x_1, x_2, …, x_n be a set of integers, and let $S = \Sigma_{i=1}^{n} x_i$. Design an algorithm to partition the set into two subsets of equal sum, or determine that it is impossible to do so. The algorithm should run in time $O(nS)$.
Solution.(Jen-Feng Shih)

Algorithm Partition_into_Two_Subsets(x);

begin
  $sum := \Sigma_{i=1}^{n} x_i$;

if sum is odd **then** print “no solution”;
else
$K := sum/2$;
$Knapsack(x, K)$;
if $P[n, K].exist = false$ **then**
print “no solution.”;
else
$l := 1$;
$m := 1$;
for $i := n$ **to** 1 **do**
if $P[i, k].belong = true$ **then**
$set1[l] := x[i]$;
$l := l + 1$;
$k := k - x[i]$;
else
$set2[m] := x[i]$;
$m := m + 1$;
print “set1:”;
for $i := 1$ **to** $l - 1$ **do**
print $set1[i]$;
print “set2:”;
for $i := 1$ **to** $m - 1$ **do**
print $set2[i]$;
end

The complexity remains the same as in the Knapsack problem, which is $O(nK) = O(nS)$.
$\square$