CSI 604 – Spring 2016
Pseudocode for some Dynamic Programming Algorithms

Handout 1.1

1. Matrix-Chain Multiplication:

The problem is to compute the matrix chain product $A_1 A_2 \cdots A_n$, where A_i is a $p_{i-1} \times p_i$ matrix for $1 \leq i \leq n$, using a minimum number of scalar multiplications. Recall that the dynamic programming approach uses an $n \times n$ array M, where $M[i, j]$, $1 \leq i \leq j \leq n$, represents the minimum number of scalar multiplications needed to compute the product of the subchain $A_i A_{i+1} \cdots A_j$. After computing all the entries of M, the solution value is given by $M[1, n]$.

```
1.  for i = 1 to n do
       M[i,i] = 0      /* Subscript difference = 0. */

2.  for q = 1 to n-1 do    /* q represents the subscript difference. */
      2.1  for i = 1 to n-q do
             2.1.1  j = i + q
             2.1.2  M[i,j] = Minimum over i <= k < j of
                             { M[i,k] + M[k+1,j] + p_{i-1} p_k p_j }

3.  Print M[1,n].   /* Optimal solution value. */
```

---

2. Maximum Subarray Sum:

Recall that in this problem, we are given a sequence S of n numbers $\langle x_1, x_2, \ldots, x_n \rangle$, some of which may be negative. The goal is to find the maximum sum over all subarrays of S.

The approach is to use an auxiliary array $B[1 \,..\, n]$ that stores the solutions to the subproblems. In particular, $B[i]$ stores the best subarray sum over all subarrays ending at x_i, $1 \leq i \leq n$.

```
1. B[1] = x_1

2. for i = 1 to n-1 do
     if (B[i] > 0)
       then B[i+1] = B[i] + x_{i+1}
       else B[i+1] = x_{i+1}

3. Print the largest value in array B.
```

(over)

3. Subset Sum Problem:

Recall that in this problem, we are given a multi-set $S = \{a_1, a_2, \ldots, a_n\}$ of positive integers and a positive integer B. The question is whether there is a subset S' of S such that the sum of the elements in S' is equal to B.

As mentioned in the lecture slides, the solution approach uses a two-dimensional Boolean array T with n rows (numbered 1 through n) and $B + 1$ columns (numbered 0 through B). The entry $T[i, j]$ is `true` if there is a subset of $\{a_1, \ldots, a_i\}$ whose sum is exactly j and `false` otherwise. After computing all the entries of T, the answer to the problem is given by $T[n, B]$.

Note: In the following pseudocode, it is assumed that if any index used for the array T is negative, then the corresponding stored value is `false`.

```
1. for j = 0 to B do   /* Initialize the first row of T. */
     if ((j = 0) or (j = a_1))
        then T[1,j] = true
        else T[1,j] = false.

2. for i = 2 to n do  /* Compute entries in row i using entries in row i-1. */
     2.1. for j = 0 to B do
             if ((T[i-1, j] = true) or (T[i-1, j-a_i] = true))
                then T[i,j] = true
                else T[i,j] = false.

3. if (T[n,B] = true)
      then print "Yes. There is a solution."
      else print "No.  There is no solution."
```
