Using dynamic programming, find the optimum printing of the text “Not all those who wander are lost”, i.e. \( \ell_1 = 3, \ell_2 = 3, \ell_3 = 5, \ell_4 = 3, \ell_5 = 6, \ell_6 = 3, \ell_7 = 4 \), with line length \( L = 14 \) and penalty function \( P(x) = x^3 \). Will the optimal printing you get be consistent with the strategy “print the word on as long as it fits, and otherwise start a new line”? Once again, you have to actually find the alignment, as opposed to only finding its penalty.

**Solution:** *************** INSERT YOUR SOLUTION HERE ***************

\[ \square \]
(a) (8 points) You are given \(n\) integers \(a_1, \ldots, a_n \geq 0\), and a target \(T \geq 0\). Design an \(O(nT)\) algorithm to determine if there exists a subset of the \(a_i\)'s that sum to \(T\). For example, if \(n = 5\) and \(a_1 = 3, a_2 = 5, a_3 = 2, a_4 = 11, a_5 = 3\), then the answer is \(YES\) for \(T = 10\) (e.g., \(3 + 5 + 2 = 10\)), but \(NO\) for \(T = 9\).

Solution: ******************* INSERT YOUR SOLUTION HERE ******************* 

(b) (4 points) Solve part (a) using only \(T\) bits of extra memory (in addition to the \(a_i\)'s themselves).

Solution: ******************* INSERT YOUR SOLUTION HERE *******************
Imagine a unary alphabet with a single letter $x$. A (valid) bracketing $B$ is a string over three symbols $x, (, )$ defined recursively as follows: (1) a single letter $x$ is a bracketing, and (2) for any $k \geq 2$, if $B_1, \ldots, B_k$ are (valid) bracketings, then so is $B = (B_1B_2\ldots B_k)$. A bracketing $B$ is called binary if rule (2) can only be applied with $k = 2$. Then the length $n$ of $B$ is the number of $x$’s it has (i.e., one ignores the parenthesis).

For example, there are 11 possible bracketings of length $n = 4$: $(xxxx), ((xx)x), ((xx)x), (x(xx)), (x(xx)), ((xx)x), (x(x(xx))), ((x(xx))x), (x((xx))x), (((xx)x)x)$, of which only the last five are binary.

(a) (4 points) Let $b(n)$ denote the number of binary bracketings of length $n$. Show that $b(n)$ is given by the following recurrence:

$$b(n) = \sum_{i=1}^{n-1} b(i)b(n-i).$$

**Solution:** **************** INSERT YOUR SOLUTION HERE ************ **

(b) (4 points) Use the result from part (a) to give a dynamic programming algorithm to compute $b(n)$ given $n$ as input. What is the running time of your algorithm? Assume that multiplication of two integers takes time $O(1)$.

**Solution:** **************** INSERT YOUR SOLUTION HERE ************ **

(c) (7 points (Extra credit)) Generalize part (a) and (b) by giving a similar recurrence(with proof) as part (a) to find the total number $f(n)$ of bracketings of length $n$, and then give a dynamic programming algorithm to compute $f(n)$ and analyze its running time.

**Solution:** **************** INSERT YOUR SOLUTION HERE ************ **
Let $*: \{1, \ldots, k\} \times \{1, \ldots, k\} \mapsto \{1, \ldots, k\}$ be a binary operation. Below we assume the values of $a*b$ for $a, b \in \{1, \ldots, k\}$ are stored in some $k \times k$ array $M$ such that $M[a][b] = a*b$. Consider the problem of examining a string $x = x_1x_2\ldots x_n$, where each $x_i \in \{1, \ldots, k\}$, and deciding whether or not it is possible to parenthesize the expression $x_1*x_2*\ldots*x_n$ in such a way that the value of the resulting expression is a given target element $t \in \{1, \ldots, k\}$. Notice, the multiplication table is neither commutative or associative, so the order of multiplication matters (and, hence, the result of the expression is not even well defined unless a complete “parenthesization” is specified). For example, consider the following multiplication table and the string $x = 2221$.

$$
\begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 1 & 3 & 3 \\
2 & 1 & 1 & 2 \\
3 & 3 & 3 & 3 \\
\end{array}
$$

Parenthesizing it $(2*2)*(2*1)$ gives $t = 1$, but $((2*2)*2)*1$ gives $t = 3$. On the other hand, no possible parenthesization gives $t = 2$ (you may check this).

(a) (8 points) Assume you are given as input the following: $n, k, t, x[1\ldots n]$ and $M$. Give a dynamic programming algorithm that runs in time polynomial in $n$ and $k$ and outputs YES if there exists a paranthesization for $x$ that results in the product equal to $t$, and NO otherwise. For instance, in the above example with $x = 2221$, the answer is YES if $t = 1$ or $t = 3$, but NO if $t = 2$.

**Solution:** ****************** INSERT YOUR SOLUTION HERE ******************

(b) (3 points (Extra credit)) Analyze the running time of your algorithm.

**Solution:** ****************** INSERT YOUR SOLUTION HERE ******************