Instructions: Please make sure that your solutions are well-thought-out, complete, concise, rigorous, and clearly written. Each solution sheet should be marked with your name, the course number, the homework number, the problem number and the date. You are allowed to discuss problems and exchange solution ideas with other students, but you must write up your own solutions. If you collaborated with other students on a problem, please acknowledge them in your write up. These are the fun problems to solve. Some of them require a bit of thinking. Please start early!

Problem 1: Alice’s RSA secret key compromised

In the RSA cryptosystem, Alice’s public key \((N, e)\) is available to everyone. Suppose that her private key \(d\) is compromised and becomes known to Eve. Show that if \(e = 3\) (a common choice) then Eve can efficiently factor \(N\).

Problem 2: Edit distance

The edit distance between two strings is defined as the minimum number of edits required to transform one string into another. An edit is one of three operations:

- **INSERT**\((x, i, a)\): This operation inserts the letter \(a\) before position \(i + 1\) in the string \(x\). Call this operation I.
- **DELETE**\((x, i)\): This operation deletes the letter at position \(i\) in the string \(x\). Call this operation D.
- **CHANGE**\((x, i, a)\): It replaces the letter at position \(i\) in the string \(x\) by the letter \(a\). Call this operation C.

For example, the string \texttt{dabbbac} can be transformed into the string \texttt{abbade} by the following edit operations:

DELETE 1
CHANGE 7 d
INSERT 7 e

Design a dynamic programming algorithm that finds the edit distance between two input strings of lengths \(m\) and \(n\), and prints an optimal sequence of edit operations. Analyze the running time and space requirements of your algorithm. To get full credit you must identify the subproblem (optimal substructure), give a recursive formulation, describe your algorithm clearly (pseudocode is preferred), and analyze the time and space complexities.

Problem 3: Palindromic subsequence

A subsequence is *palindromic* if it is the same whether read left-to-right or right-to-left. For instance, the sequence

\[
\]

has many palindromic subsequences, including \(A, C, G, C, A\) and \(A, A, A, A\) (on the other hand, the subsequence \(A, C, T\) is not palindromic). Devise an algorithm which takes a sequence \(x[1 \ldots n]\) and returns the (length of the) longest palindromic subsequence. Its running time should be \(O(n^2)\). You
must clearly show the subproblem(s) you are solving, give a recursive formulation, give pseudocode for the algorithm, analyze the running time and space.

**Problem 4: 3-Partition**  
(8 + 8 + 5 + 4 = 25 Points)
Consider the following 3-PARTITION problem. Given integers $a_1, \ldots, a_n$, we want to determine whether it is possible to partition \{1, \ldots, n\} into three disjoint subsets $I, J$ and $K$ such that

$$
\sum_{i \in I} a_i = \sum_{j \in J} a_j = \frac{1}{3} \sum_{i=1}^{n} a_i = \frac{1}{3} S.
$$

For example, for input (1, 2, 3, 4, 5, 8) the answer is yes, because there is the partition (1, 8), (4, 5), and (2, 3, 4). On the other hand, for input (2, 2, 3, 5) the answer is no.

Devise and analyze a dynamic programming algorithm for 3-PARTITION that runs in time polynomial in $n$ and in $S$. You must clearly show the subproblem(s) you are solving, give a recursive formulation, give pseudocode for the algorithm, analyze the running time and space.

**Problem 5: Add up to the target**  
(8 + 8 + 5 + 4 = 25 Points)
Give an algorithm for the following task.

*Input:* A list of $n$ positive integers $a_1, \ldots, a_n$; a positive integer $t$. Each of the $a_i$’s can be used any number of times to construct $t$.

*Question:* Does some subset of the $a_i$’s (with repetitions allowed) add up to $t$?

You must clearly show the subproblem(s) you are solving, give a recursive formulation, give pseudocode for the algorithm, analyze the running time and space.