1. Describe and analyze an algorithm to determine whether a given set of positive integers can be partitioned into three disjoint subsets whose sums are equal. That is, given a set X whose elements sum to S, your algorithm should determine whether there are three disjoint subsets $A, B, C \subseteq X$ such that $A \cup B \cup C = X$ and

$$\sum_{a \in A} a = \sum_{b \in B} b = \sum_{c \in C} c = \frac{S}{3}.$$

For full credit, your algorithm should run in time polynomial in S and n.

For example, the set $\{2, 3, 4, 6, 7, 8, 9, 12\}$, can be partitioned into the subsets $\{2, 3, 12\}$, $\{4, 6, 7\}$, and $\{8, 9\}$, each of whose elements sum to 17. On the other hand, there is no balanced partition of the set $\{4, 8, 15, 16, 23, 42\}$ into three subsets.

2. Suppose Scrooge McDuck wants to split a gold chain into $n + 1$ smaller chains to distribute to his numerous nephews and nieces as their inheritance. Scrooge has carefully marked n locations where he wants the chain to be cut.

McDuck's jeweler, Fenton Crackshell, agrees to cut any chain of length ℓ inches into two smaller pieces, at any specified location, for a fee of ℓ dollars. To cut a chain into more pieces, Scrooge must pay Crackshell separately for each individual cut. As a result, the cost of breaking the chain into multiple pieces depends on the order that Crackshell makes his cuts. Obviously, Scrooge wants to pay Crackshell as little as possible.

For example, suppose the chain is 12 inches long, and the cut locations are 3 inches, 7 inches, and 9 inches from one end of the chain. If Crackshell cuts first at the 3-inch mark, then at the 7-inch mark, and then at the 9-inch mark, Scrooge must pay $12 + 9 + 5 = 26$ dollars. If Crackshell cuts the chain first at the 9-inch mark, then at the 7-inch mark, and then at the 3-inch mark, Scrooge must pay $12 + 9 + 7 = 28$ dollars. Finally, if Crackshell makes his first cut at the 7-inch mark, Scrooge only owes him $12 + 7 + 5 = 24$ dollars.

Describe and analyze an efficient algorithm that computes the minimum cost of partitioning the chain. The input to your algorithm is a sorted array $C[1..n+1]$, where $C[1]$ through $C[n]$ are the n cut positions, and $C[n+1]$ is the total length of the chain.

Rubric: For *all* dynamic programming problems in this class:

- 60% for a correct recurrence, including base cases and a plain-English description of the function. No credit for anything else if this is wrong.
- 10% for describing a suitable memoization data structure.
- 20% for describing a correct evaluation order. A clear picture is sufficient.
- 10% for analyzing the resulting algorithm's running time. It is not necessary to state a space bound.

Official solutions will always include pseudocode for the final iterative dynamic programming algorithm, but this is ***not*** required for full credit. On the other hand, if you do provide correct pseudocode for the iterative algorithm, you do not need to ***separately*** describe the recurrence, the memoization structure, or the evaluation order.