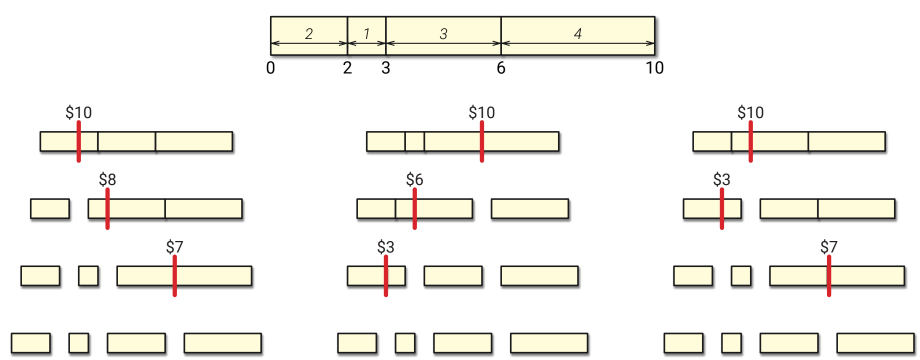
Write your answers in the separate answer booklet.
Please return this question sheet and your cheat sheet with your answers.

1. Suppose you want to build a house for your new dog Fluffy. Your friend, who happens to be a master carpenter, has given you a long plank of wood, helpfully marked at $n-1$ places where you should cut the plank into the n shorter boards you actually need to build Fluffy's doghouse. Since you don't own a saw, and your friend is out of town, you drive your plank down to Woodchuck & Woodchuck Woodcutters. Woodchuck & Woodchuck charge more to cut longer planks, at a rate of \$1 per foot. For example, they charge \$13 to cut a 13-foot-long plank, but only \$1.50 to cut a 1½-foot-long plank.

   The total price for all $n-1$ cuts depends on the order in which the cuts are made. For example, suppose your plank is 10 feet long, and the marks are 2 feet, 3 feet, and 6 feet from the left end of the plank, as illustrated below.

   - Making the cuts in order from left to right costs $\$10+\$8+\$7=\$25$.
   - Making the cuts in order from right to left costs $\$10+\$6+\$3=\$19$.
   - Making the middle cut first and then the other two costs $\$10+\$3+\$7=\$20$.



   Describe and analyze an efficient algorithm that returns the minimum cost to make all $n-1$ marked cuts. The input to your algorithm is a sorted array $M[1..n]$ of positive numbers, where $M[i]$ is the distance from the left end of the plank to the ith cut mark, and $M[n]$ is the total length of the plank.

2. Describe and analyze an efficient algorithm to find the length of the longest substring that appears both forward and backward in an input string $T[1..n]$. The forward and backward substrings must not overlap. Here are several examples:

   - Given the input string `ALGORITHM`, your algorithm should return 0.
   - Given the input string `RECURSION`, your algorithm should return 1, for the substring `R`.
   - Given the input string `REDIVIDE`, your algorithm should return 3, for the substring `EDI`. (Remember: The forward and backward substrings must not overlap!)
   - Given the input string `SOMANYDYNAMICPROGRAMS`, your algorithm should return 4, for the substring `MANY`.

3. Suppose you are given a directed acyclic graph G whose nodes represent jobs and whose edges represent precedence constraints; that is. each edge $u \rightarrow v$ indicates that job u must be completed before job v begins. Each node v stores a non-negative number $v.duration$ indicating the time required to execute job v. All jobs are executed in parallel; any job can start or end while any number of other jobs are executing, provided all the precedence constraints are satisfied. You'd like to get all these jobs done as quickly as possible.

   Describe an algorithm to determine, for every vertex v in G, the earliest time that job v can ***begin***, assuming no job starts before time 0 and no precedence constraints are violated. Your algorithm should record the answer for each vertex v in a new field $v.earliest$.

4. Oh, no! You have been appointed as the gift czar for Twitbook's annual mandatory holiday party! The president of the company has declared that every Twitbook employee must receive one of three gifts: (1) an all-expenses-paid six-week vacation anywhere in the world, (2) an all-the-pancakes-you-can-eat breakfast for two at Jumping Jack Flash's Flapjack Stack Shack, or (3) a burning paper bag full of dog poop. Company rules prohibit any employee from receiving the same gift as their direct supervisor. Any employee who receives a *better* gift than their direct supervisor will almost certainly be fired in a fit of jealousy. How do you choose gifts so that as few people as possible get fired?

   More formally, suppose you are given a rooted tree T, representing the Twitbook company hierarchy. You need to label each vertex of T with an integer 1, 2, or 3, such that every node has a different label from its parent. The *cost* of a labeling is the number of vertices that have smaller labels than their parents. Describe and analyze an algorithm to compute the minimum cost of any labeling of the given tree T.

   For example, the following figure shows a tree labeling with cost 9; the nine bold nodes have smaller labels than their parents. (This is *not* the optimal labeling for this tree.)