You have 120 minutes to answer five questions.

Write your answers in the separate answer booklet.

Please return this question sheet and your cheat sheet with your answers.

1. Short answers:

   (a) Solve the following recurrences:

   - $A(n) = A(5n/11) + O(\sqrt{n})$
   - $B(n) = 8B(n/2) + O(n^2)$
   - $C(n) = C(n/2) + C(n/3) + C(n/6) + O(n)$

   (b) Describe an appropriate memoization structure and evaluation order for the following (meaningless) recurrences, and state the running time of the resulting iterative algorithm to compute the requested function value.

   - Compute $Foo(1, n)$ where

$$Foo(i,k) = \begin{cases} 0 & \text{if } i \geq k-1 \\ \max\left\{ \begin{array}{c} Foo(i,j) \\ +\, Foo(j,k) \end{array} \;\middle|\; i < j < k \right\} + \sum_{j=i}^{k} A[j] & \text{otherwise} \end{cases}$$

   - Compute $Bar(n, 1)$ where

$$Bar(i,s) = \begin{cases} \infty & \text{if } i < 0 \text{ or } s > n \\ 0 & \text{if } i = 0 \\ \min\left\{ \begin{array}{c} Bar(i, 2s), \\ X[i] \cdot s + Bar(i-s, s) \end{array} \right\} & \text{otherwise} \end{cases}$$

2. Chandler is moving to Tulsa, Oklahoma, to start a new job, and he wants to plan a walk from home to work. He wants to continue his habit of buying a ~~hot milkshake~~ salted caramel latte from a local coffee shop and a paper from a local newsstand every morning. (Yes, an actual *paper* paper. Could he *be* any more retro?)

   Chandler has a map of his new neighborhood in the form of an undirected graph G, whose vertices represent intersections and whose edges represent roads between them. Every edge e has a positive length $\ell(e)$. A subset of the vertices are marked as newsstands; another disjoint subset of vertices are marked as coffee shops. The graph has two special vertices s and t, which represent Chandler's home and work, respectively.

   Describe an algorithm that computes the shortest route that Chandler can follow from home to work that visits both a coffee shop and a newsstand, or correctly reports that no such route exists (which means Chandler should move back to New York).

Problems 3 and 4 are on the back of this page.

3. Recall that an *arithmetic progression* is any sequence of real numbers $x_1, x_2, \ldots, x_n$ such that $x_{i+1} - x_i = x_i - x_{i-1}$ for every index $2 \le i \le n-1$.

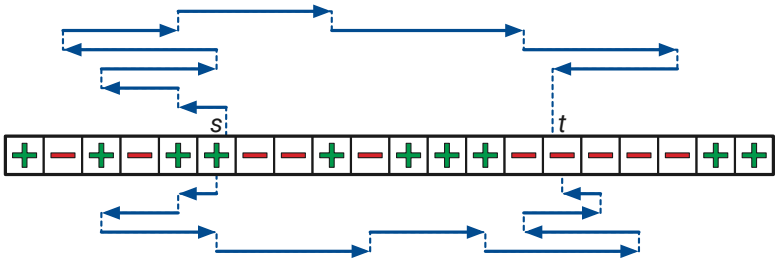
   Suppose we are given a sorted array $X[1..n]$ that contains an arithmetic sequence ***with one element deleted***. Describe and analyze an algorithm to find the deleted element as quickly as possible. (If there are multiple correct answers, your algorithm can return any one of them.)

   For example, given the input array $X = [2, 4, 8, 10, 12]$, your algorithm should return 6, and given the array $X = [21, 18, 15, 12]$, your algorithm should return either 9 or 24.

4. ***Ink-deck*** is a solitaire puzzle game played on a row of n squares, each marked with either $+$ or $-$. Your goal is to move a token from a specified start square s to a specified target square t using a sequence of *moves*. Each move translates the token either left or right along the row to a new square. The *length* of a move is the distance that the token moves; for example, a move from square 5 to square 12 has length 7. Moves are subject to the following rules:

   - The first move must have length 1.
   - If the token is on a square marked $+$, your next move must be one square longer than your previous move.
   - If the token is on a square marked $-$, your next move must be one square shorter than your previous move. (In particular, if your previous move had length 1, then you cannot move at all!)
   - You are never allowed to move the token off either end of the row.

   The following figure shows an example ink-deck puzzle, along with two solutions (which might not be optimal).



An ink-deck puzzle with two nine-move solutions.

   (a) Describe an algorithm that either finds a solution *with the minimum number of moves* for a given ink-deck puzzle, or correctly reports that the given puzzle has no solution.

   (b) Describe an algorithm that either finds a solution *whose final move is as long as possible* for a given ink-deck puzzle, or correctly reports that the given puzzle has no solution.

   Your input to both algorithms consists of an array $ID[1..n]$, where $ID[i] \in \{-1, +1\}$ for each index i, along with two indices $1 \le s \le n$ and $1 \le t \le n$.

Problem 5 is on the next page.

5. Suppose you are given a string of symbols, representing a message in some foreign language that you do not understand, in an array $T[1..n]$. You have access to a black-box subroutine IsWord that takes a string w as input and decides in $O(|w|)$ time whether w is a word.

   You eagerly implement and run the text-splitting algorithm we saw in class, only to discover that the given string *cannot* be split into words! Apparently, as a crude form of cryptography, the message has been corrupted by adding extra symbols between words.

   So you decide instead to look for as many non-overlapping words in T as possible. A *verbal subsequence* of T is a sequence of non-overlapping substrings of T, each of which is a word. The *length* of a verbal subsequence is the number of words it contains. Describe and analyze an algorithm to find the length of the longest verbal subsequence of a given string T.

   For example, suppose IsWord(w) returns True if and only if w is an English word with at least four letters. Then (STUDY, MICE, TRAP, RAMEN) and (DYNAMIC, EXTRA, PROGRAM) are verbal subsequences of the string STUDYNAMICEXTRAPROGRAMEN:

   Given the input string STUDYNAMICEXTRAPROGRAMEN, your algorithm should return the integer 4, which is the length of the verbal subsequence (STUDY, MICE, TRAP, RAMEN).

Nothing to see here.