Write your answers in the separate answer booklet.

You have 120 minutes (after you get the answer booklet) to answer five questions.

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.

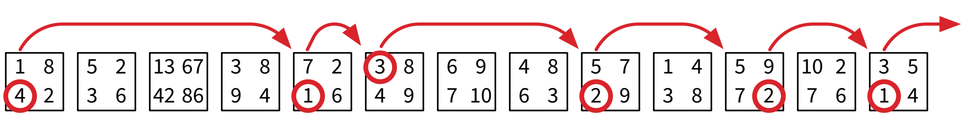
   - Compute $\mathit{Foo}(1, n)$ where

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

   - Compute $\mathit{Bar}(n, 1)$ where

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

2. *Quadhopper* is a solitaire game played on a row of n squares. Each square contains four positive integers. The player begins by placing a token on the leftmost square. On each move, the player chooses one of the numbers on the token's current square, and then moves the token that number of squares to the right. The game ends when the token moves past the rightmost square. The object of the game is to make as many moves as possible before the game ends.



A quadhopper puzzle that allows six moves. (This is ***not*** the longest legal sequence of moves.)

   (a) ***Prove*** that the obvious greedy strategy (always choose the smallest number) does not give the largest possible number of moves for every quadhopper puzzle.

   (b) Describe and analyze an efficient algorithm to find the largest possible number of legal moves for a given quadhopper puzzle.

3. After moving to a new city, you decide to walk from your home to your new office. To get a good daily workout, you want to reach the highest possible altitude during your walk (to maximize exercise), while keeping the total length of your walk below some threshold (to get to your office on time). Describe and analyze an algorithm to compute the best possible walking route.

   Your input consists of an undirected graph G, where each vertex v has a height $h(v)$ and each edge e has a positive length $\ell(e)$, along with a start vertex s, a target vertex t, and a maximum length L. Your algorithm should return the maximum height reachable by a walk from s to t in G, whose total length is at most L.

4. 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 can decide whether an arbitrary string w is a word in $O(|w|)$ time.

   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 a common English word. Then (`STUDY`, `AM`, `ICE`, `TRAP`, `RAMBLE`) and (`DYNAMIC`, `EXTRA`, `PROGRAM`) are verbal subsequences of the string `STUDYNAMICEXTRAPROGRAMBLE`:

   $$\boxed{\texttt{STUDY}}\texttt{N}\boxed{\texttt{AM}}\,\boxed{\texttt{ICE}}\texttt{X}\boxed{\texttt{TRAP}}\texttt{ROG}\boxed{\texttt{RAMBLE}} \qquad \texttt{STU}\boxed{\texttt{DYNAMIC}}\boxed{\texttt{EXTRA}}\boxed{\texttt{PROGRAM}}\texttt{BLE}$$

   Thus, given the input string `STUDYNAMICEXTRAPROGRAMBLE`, the output of your algorithm should be *at least* 5, which is the length of the subsequence (`STUDY`, `AM`, `ICE`, `TRAP`, `RAMBLE`). (This string may contain longer verbal subsequences.)

5. 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. To avoid annoying boundary cases, you can assume $n \ge 4$.

   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.