

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. (a) Solve the following recurrences:

- $A(n) = 4A(n/9) + O(\sqrt{n})$
- $B(n) = 2B(n/3) + B(n/6) + O(n)$
- $C(n) = 9C(n/3) + O(n^2)$

- (b) After making it most of the way through solving a complicated dynamic programming problem, you've come up with the following recurrence. Here, $A[1..n]$ is an array holding n arbitrary numbers.

$$BestFlobble(i, k) = \begin{cases} 0 & \text{if } i \geq k \\ \min \left\{ \begin{array}{l} BestFlobble(i, j-1) \\ + (j-i) \cdot A[j] \\ + BestFlobble(j, k) \end{array} \middle| i < j \leq k \right\} & \text{otherwise} \end{cases}$$

The solution to the problem can be obtained by evaluating $BestFlobble(1, n)$. Describe an appropriate memoization structure and evaluation order for computing that function value, and state the running time of the resulting iterative algorithm.

2. Consider the following context-free grammar (CFG) G :¹

$$S \rightarrow 0S1 \mid 1S0 \mid \varepsilon$$

Recall, G **generates** a string $w \in \{0, 1\}^*$ if (informally) w can be obtained by starting with the *string* S and repeatedly transforming it by replacing the non-terminal symbol S with one of the strings to the right of the above arrow. For example, G generates 101010 through the following transformations:

$$S \rightsquigarrow 1S0 \rightsquigarrow 10S10 \rightsquigarrow 101S010 \rightsquigarrow 101010$$

Describe an algorithm that given a string $w[1..n] \in \{0, 1\}^*$ computes the length of the longest *subsequence* of w that is generated by G .

¹The version of this exam originally prepared for distribution to students used the grammar $S \rightarrow 00S1 \mid 1S0 \mid \varepsilon$. While there is a solution with the same running time using the grammar in this footnote, it makes the problem quite a bit more complicated than intended.

3. You're visiting a friend in a new city with three overlapping tram systems AirTram, BestTram, and CarryTram (or Systems A, B, and C for short) that share the same stations but offer different connections between them. All three systems charge money to use their connections but offer a discount to card-carrying members of that system.

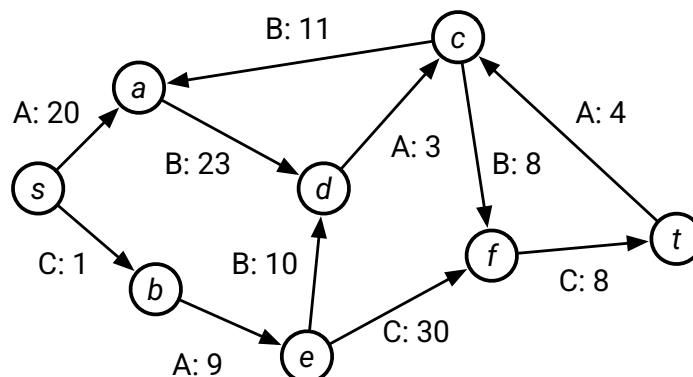
Your friend had offered to hook you up with, "a membership card for all three tram systems," but when you first arrived at his home, he revealed that he really had a single card for System C with two stickers stacked on top, each designed to make it look like a card for the one of the other systems. The effects of this scheme can be summarized as follows:

- As long as you don't remove either sticker, you'll appear to be a member of System A.
- Once you remove one sticker, you'll appear to be a member of system B.
- Once you remove the second sticker, you'll appear to be a member of System C.
- You'll pay **half** the cost of using a connection run by the system you currently appear to be a member of.

The stickers were produced with cheap adhesive, so neither of them can be reapplied once they are removed. However, nobody at the tram stations will be watching you very closely, so you can remove one or both stickers at any stop including the one at your friend's house. **Your goal is to get from your friend's house to the Transit Authority as cheaply as possible**, so you can buy yourself some *proper* membership cards.

Formally, you are given a directed graph $G = (V, E)$ whose n vertices represent tram stations and whose m edges represent direct connections between pairs of stations. Every edge e is given a label $system(e) \in \{A, B, C\}$ saying whether that connection is run by System A, System B, or System C, respectively, along with a positive cost $\$(e)$ for **non-members** to use that connection (again, you'll pay half that cost if you appear to be a member of the connection's system.)

Describe an algorithm to compute the minimum cost to get from your friend's house at vertex s to the Transit Authority at vertex t . For example, given the graph below with each edge labeled by its system and cost, your algorithm should return 21.5 for the plan of traveling along $s \rightarrow b \rightarrow e$ while appearing to be a member of System A at cost $1 + 9/2 = 5.5$, removing one sticker, traveling along $e \rightarrow d \rightarrow c \rightarrow f$ while appearing to be a member of System B at cost $10/2 + 3 + 8/2 = 12$, removing the second sticker, and traveling along $f \rightarrow t$ while appearing to be a member of System C at cost $8/2 = 4$.



4. Suppose you are given a sorted array of n distinct numbers that has been rotated *forward* k steps for some **unknown** integer k between 1 and $n - 1$. In other words, you are given an array $A[1..n]$ such that some prefix $A[1..k]$ is sorted in increasing order, the complementary suffix $A[k + 1..n]$ is sorted in increasing order, and $A[n] < A[1]$.

For example, you might be given the following 16-element array.

9	13	16	18	19	23	28	31	37	42	1	3	4	5	7	8
---	----	----	----	----	----	----	----	----	----	---	---	---	---	---	---

In this case, we can determine that $k = 10$.

Describe an algorithm to determine if the given array A contains a given number x . For example, given the array above and $x = 19$ your algorithm should return `TRUE`, but given the array above and $x = 6$, your algorithm should return `FALSE`.

5. Consider the following solitaire (single-player) card game. You begin with a sequence of $n \geq 1$ cards laid out on a table from left to right. Each card has a non-negative integer written on it. At the beginning of each turn, you are either **holding** a card, or your hand is **empty**. In particular, your hand is empty at the beginning of your first turn.

Each turn proceeds as follows:

- i. You pick up the leftmost card on the table. Let's call it card C .
- ii. If you *are not* holding a card (your hand is empty), you have the choice to either **discard** card C by tossing it into a nearby trash bin or to hold card C .

However, if you *are* holding a card H , you have the choice to either add the product of H and C 's numbers to your score before discarding both cards (emptying your hand) or to merely discard C and continue holding H .

The game ends when you begin a turn with no cards remaining on the table. If you are holding a card H , then add ten times the number on H to your score. Otherwise, your score remaining unchanged. You begin the game with a score of 0.

Describe an algorithm to compute the maximum score you can achieve over the course of one game. You may assume the input to your algorithm is an array $Cards[1..n]$ where $Cards[i]$ is the number written on the i th card from the left of those initially on the table.

For example, given the sequence of 6 cards with the numbers shown below, your algorithm should return 182 for the strategy of discarding the 2 card, holding the 8 card, discarding the 3 card, adding $8 \cdot 9$ to the score before discarding both cards, holding the 11 card, discarding the 1 card, and ending the game by adding $10 \cdot 11$ to the score.

2	8	3	9	11	1
---	---	---	---	----	---