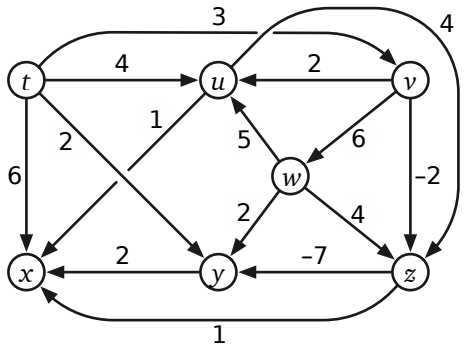
Write your answers in the separate answer booklet.

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. ***Clearly*** indicate the following structures in the directed graph below. Don't be subtle! To indicate a subset of edges, draw a **HEAVY BLACK LINE** along the entire length of each edge. If the requested structure does not exist, write the word NONE. (The answer booklet contains several copies of this graph.)



   (a) A depth-first search tree rooted at v

   (b) A breadth-first search tree rooted at w

   (c) A shortest-path tree rooted at t

   (d) A list of vertices in topological order

2. Suppose you are given a directed graph $G = (V, E)$, each of whose edges are colored red, green, or blue. Edges in G do not have weights, and G is not necessarily a dag. A *rainbow walk* is a walk in G that does *not* contain two consecutive edges with the same color.

   Describe and analyze an algorithm to find all vertices in G that are reachable from a given vertex s through a rainbow walk.

3. Suppose we are given an n-digit integer X. Repeatedly delete one digit from either end of X (your choice) until no digits are left. The *square-depth* of X is the maximum number of perfect squares that you can see during this process.

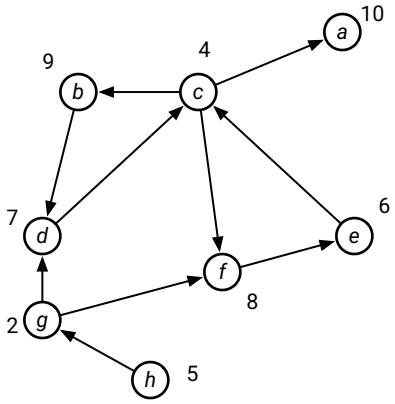
   For example, the integer 32492 has square-depth 3, by the following sequence of digit deletions:

$$32492 \xrightarrow{57^2} 3249 \xrightarrow{18^2} 324 \xrightarrow{2^2} 24 \to 4 \to \varepsilon.$$

   Describe and analyze an algorithm to compute the square-depth of a given integer X, represented as an array $X[1..n]$ of n decimal digits. Assume you have access to a subroutine IsSquare that determines whether a given k-digit number (represented by an array of digits) is a perfect square ***in $O(k^2)$ time***.

4. Suppose you are given k arrays $A_1[1..n]$, $A_2[1..n]$, ..., $A_k[1..n]$, each with the same length n, and each sorted in increasing order. Describe an algorithm to merge the given arrays into a single sorted array. Analyze the running time of your algorithm as a function of n and k.

5. Let $G = (V, E)$ be a directed graph with vertex weights $w : V \to \mathbb{R}$. Given a vertex u, let $\mathit{MinReach}(u) := \min\{w(v) \mid u \text{ can reach } v\}$ be the minimum weight over all vertices v such that u can reach v. For example, if G is the graph shown below, then $\mathit{MinReach}(f) = 4$ and $\mathit{MinReach}(h) = 2$.



   (a) Suppose G is a directed acyclic graph (dag). Describe an algorithm to compute $\mathit{MinReach}(u)$ for *every* vertex $u \in V$.

   (b) Extend your algorithm to the case of a general directed graph. You may use the algorithm from part (a) as a black-box whether or not your solution for part (a) is correct.