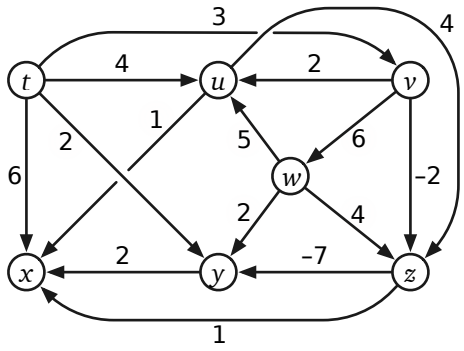
Write your answers in the separate answer booklet.

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:

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

   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. You are planning a hiking trip in Jellystone National Park over winter break. You have a complete map of the park's trails; the map indicates that some trail segments have a high risk of bear encounters. All visitors to the park are required to purchase a canister of bear repellent. You can safely traverse a high-bear-risk trail segment only by *completely* using up a *full* canister of bear repellent. The park rangers have installed refilling stations at several locations around the park, where you can refill empty canisters at no cost. The canisters themselves are expensive and heavy, so you cannot carry more than one. Because the trails are narrow, each trail segment allows traffic in only one direction.

   You have converted the trail map into a directed graph $G = (V, E)$, whose vertices represent trail intersections, and whose edges represent trail segments. A subset $R \subseteq V$ of the vertices indicate the locations of the *R*epellent *R*efilling stations, and a subset $B \subseteq E$ of the edges are marked as having a high risk of *B*ears. Your campsite appears on the map as a particular vertex $s \in V$, and the visitor center is another vertex $t \in V$.

   (a) Describe and analyze an algorithm to decide if you can safely walk from your campsite s to the visitor center t. Assume there is a refill station at your camp site, and another refill station at the visitor center.

   (b) Describe and analyze an algorithm to decide if you can safely walk from any refill station any other refill station. If there a safe path from u to v for *every* pair of vertices u and v in R, your algorithm should return TRUE; otherwise, it should return FALSE.