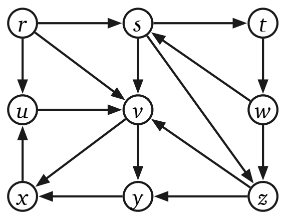
Write your answers in the separate answer booklet.
Please return this question sheet and your cheat sheet with your answers.

1. ***Clearly*** indicate the following structures in the directed graph below, or write NONE if the indicated structure does not exist. (There are several copies of the graph in the answer booklet.)



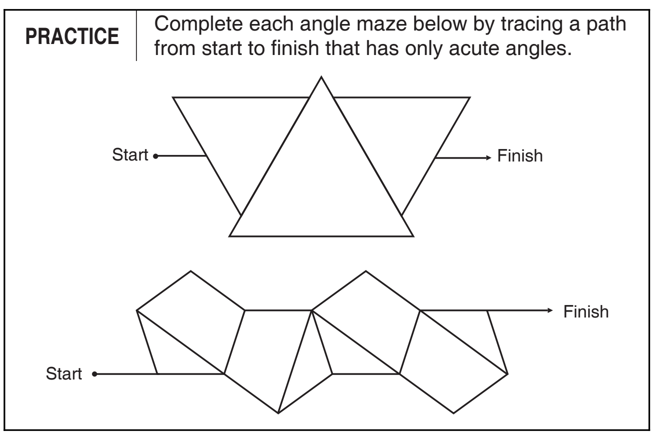
(a) A depth-first spanning tree rooted at r.

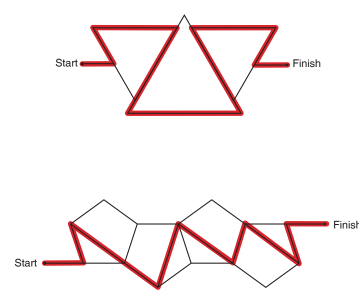
(b) A breadth-first spanning tree rooted at r.

(c) A topological order.

(d) The strongly connected components.

2. The following puzzles appear in my younger daughter's math workbook.[1] (I've put the solutions on the right so you don't waste time solving them during the exam.)





Describe and analyze an algorithm to solve arbitrary acute-angle mazes.

You are given a connected undirected graph G, whose vertices are points in the plane and whose edges are line segments. Edges do not intersect, except at their endpoints. For example, a drawing of the letter X would have five vertices and four edges; the first maze above has 13 vertices and 15 edges. You are also given two vertices Start and Finish.

Your algorithm should return TRUE if G contains a walk from Start to Finish that has only acute angles, and FALSE otherwise. Formally, a walk through G is valid if, for any two consecutive edges $u \to v \to w$ in the walk, either $\angle uvw = 180°$ or $0 < \angle uvw < 90°$. Assume you have a subroutine that can compute the angle between any two segments in $O(1)$ time. Do ***not*** assume that angles are multiples of $1°$.

[1] Jason Batterson and Shannon Rogers, *Beast Academy Math: Practice 3A*, 2012. See https://www.beastacademy.com/resources/printables.php for more examples.

3. Suppose you are given a sorted array $A[1\,..\,n]$ of distinct numbers that has been *rotated k* steps, for some ***unknown*** integer k between 1 and $n-1$. That is, the prefix $A[1\,..\,k]$ is sorted in increasing order, the 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 (where $k = 10$):

9	13	16	18	19	23	28	31	37	42	−4	0	2	5	7	8

Describe and analyze an efficient algorithm to determine if the given array contains a given number x. The input to your algorithm is the array $A[1\,..\,n]$ and the number x; your algorithm is ***not*** given the integer k.

4. You have a collection of n lockboxes and m gold keys. Each key unlocks *at most* one box; however, each box might be unlocked by one key, by multiple keys, or by no keys at all. There are only two ways to open each box once it is locked: Unlock it properly (which requires having a matching key in your hand), or smash it to bits with a hammer.

   Your baby brother, who loves playing with shiny objects, has somehow managed to lock all your keys inside the boxes! Luckily, your home security system recorded everything, so you know exactly which keys (if any) are inside each box. You need to get all the keys back out of the boxes, because they are made of gold. Clearly you have to smash at least one box.

   (a) Your baby brother has found the hammer and is eagerly eyeing one of the boxes. Describe and analyze an algorithm to determine if it is possible to retrieve all the keys without smashing any box except the one your brother has chosen.

   (b) Describe and analyze an algorithm to compute the minimum number of boxes that must be smashed to retrieve all the keys.

5. It's almost time to show off your flippin' sweet dancing skills! Tomorrow is the big dance contest you've been training for your entire life, except for that summer you spent with your uncle in Alaska hunting wolverines. You've obtained an advance copy of the the list of n songs that the judges will play during the contest, in chronological order.

   You know all the songs, all the judges, and your own dancing ability extremely well. For each integer k, you know that if you dance to the kth song on the schedule, you will be awarded exactly $Score[k]$ points, but then you will be physically unable to dance for the next $Wait[k]$ songs (that is, you cannot dance to songs $k+1$ through $k + Wait[k]$). The dancer with the highest total score at the end of the night wins the contest, so you want your total score to be as high as possible.

   Describe and analyze an efficient algorithm to compute the maximum total score you can achieve. The input to your sweet algorithm is the pair of arrays $Score[1\,..\,n]$ and $Wait[1\,..\,n]$.