NAME (1 pt): _____

TA (1 pt): _____

Name of Neighbor to your left (1 pt): _____

Name of Neighbor to your right (1 pt): _____

This is a closed book, closed calculator, closed computer, closed network, open brain exam, but you are permited a 2 page, double-sided set of notes, large enough to read without a magnifying glass.

You get one point each for filling in the 4 lines at the top of this page. Each other question is marked by the number of points it is worth. You may use the number of points assigned to each problem as a rough estimate for the number of minutes you want to allocate to the problem. The total number of points is 149. (This means that we hope you will have enough time to finish the exam!)

Write all your answers on this exam. If you need scratch paper, ask for it, write your name on each sheet, and attach it when you turn it in (we have a stapler).

Question	Score	Max
Cover		4
1		20
2		20
3		15
4		20
5		15
6		30
7		25
Total		149

1. 2s complement arithmetic. (**20 points.**)

   You are running on a computer with 10-bit 2s complement arithmetic. One integer fits in one 10-bit word.

   (a) (**5 points.**) What is the representable range of integers? I.e. what are the most negative and most positive representable integers? Give you answers in decimal and binary notation.

   (b) (**5 points.**) What is the value returned by the following computation? Express your answer as decimal integers. Which operations signal integer overflow?

   -266-246
   300+400
   -305-405
   3*400
   -4*400

   (c) (**10 points.**) What does the following program print out? All variables are 10-bit 2s complement integers. Justify your answer.

   $m = $ most negative representable integer
   $M = $ most positive representable integer
   $count = 0$
   for $i = m$ to M
       for $j = m$ to M
           if $(12 * i = 52 * j - 28)$ then $count = count + 1$
   endfor
   endfor
   print($count$)

2. Toy Train. **(20 points)**

A precocious 4-year-old M likes to play with her toy trains. There are 4 kinds of track pieces that can be connected (for convenience, each place a track piece can connect to another is labeled as in the figure below):

(a) straight pieces (one end is labeled A and the other B),

(b) curved pieces (one end is labeled A and the other B),

(c) Y-connectors, where if you enter at C, you can exit either at D or E, and if you enter at D or E, you must exit at C

(d) T-connectors, where no matter where you enter (say F), you can exit at either of the other two places (G or H)

**Straight   Curved   Y–connector   T–connector**
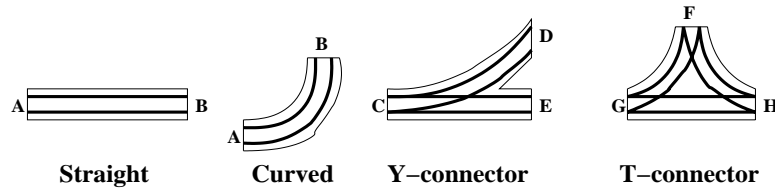
After building lots of different train tracks, M realizes not all are equally fun to play with: On some tracks you can keep moving the train forward as long as you like without running out of track, and on some you can't. On some you can get to any track piece from any other track piece by moving the train in its forward direction. On some you can even get the train to face in either direction on any track piece just by moving forward (see Track A below, where Train 1 can get to the same position as Train 2). In some other tracks you can't even reach some track pieces depending on where you start (see Track B below, where Train 3 stays on the lower track as long as it moves forward).
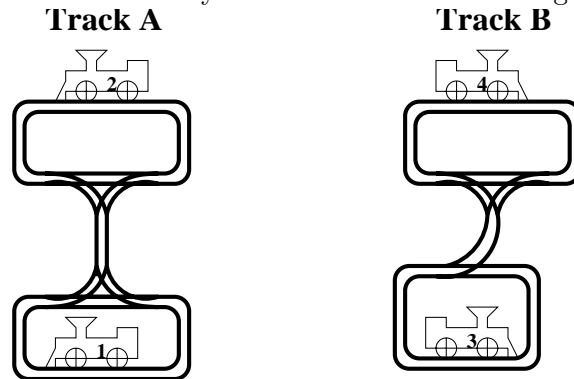
Track A   **Track B**

(a) **(5 points.)** Given a train track, we will define a graph G that describes the train track, and how trains can move if they are only allowed to move forward. There will be a pair of vertices to represent each straight or curved piece: one vertex representing a train moving forward from A to B, and the other vertex representing a train moving forward from B to A. (Label these vertices for piece i vertices iAB and iBA, respectively.) Similarly, there will be a pair of vertices for each of the 3 places a Y- or T-connector can connect to another track piece. (For example, two of the vertices for T-connector j are labeled $jFin$ and $jFout$, for trains moving toward the T-connector at F and moving away from the connector at F, respectively.) There will be edges to represent connectors, and to represent how track pieces are joined together. Show how to define these graphs by filling in the missing edges in the figures below (all possible vertices are not shown in each figure; don't worry about edges to vertices that are not shown). One edge is already filled in for you:

the edge from 1AB to 2AB means that a train on piece 1 facing from A to B can move forward to piece 2 and also face from A to B.

If there are a total of s straight or curved pieces and c Y- or T-connectors, give the exact number of vertices n and an upper bound on the number of edges e (in terms of s and c) in your graph G.
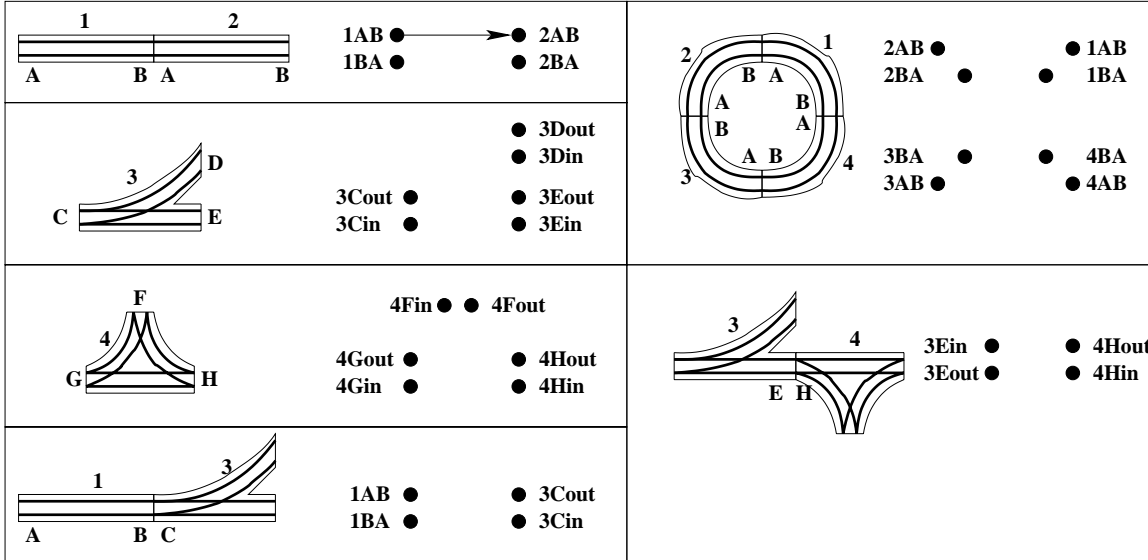
(b) **(5 points.)** Briefly describe an efficient algorithm for figuring out whether M can put a train on a chosen straight or curved track piece, and facing in a chosen direction, and move it forward to reach another chosen destination straight or curved track piece facing in a chosen destination direction. For example, Train 3 on Track B above cannot reach the position of Train 4 above moving forward only, but Train 4 can reach the position of Train 3. Similarly, Train 1 on Track A can reach the position of Train 2. Give the complexity in terms of the total number s of straight or curved pieces and the total number c of Y- or T-connectors. Justify your answer briefly.

4

(c) (**5 points.**) Briefly describe an efficient algorithm for figuring out whether M can put a train on a chosen straight or curved track piece and facing in a chosen direction, and then play as long as she wants, i.e. keep moving the train forward without running out of track, provided she makes the correct turns at connectors. Give the complexity in terms of the total number s of straight or curved pieces and the total number c of Y- or T-connectors. Justify your answer briefly.

(d) (**5 points.**) Briefly describe an efficient algorithm for answering the last question when M plays with her eyes closed after putting the train down on a chosen track piece facing in a chosen direction, i.e. may take any possible turn at any connector. Is she guaranteed to be able to keep the train moving forward as long as she likes without running out of track, regardless of what turns she makes at connectors? Give the complexity in terms of the total number s of straight or curved pieces and the total number c of Y- or T-connectors. Justify your answer briefly.

(e) (**5 points of extra credit.**) M's little brother N shows up and starts playing, but moves the train either forward or backward as he pleases. Answer question (b) again: Briefly describe an efficient algorithm for figuring out whether N can put a train on a chosen straight or curved track piece, and facing in a chosen direction, and move it forward *or backward* to reach another chosen destination straight or curved track piece facing in a chosen destination direction.

3. FFT. **15 points.**

(a) (11 points) You want to multiply two degree-n polynomials $p(x) = q(x) \cdot r(x)$ using the FFT. Explain why the following algorithm is wrong and fix it.

We let $q(x) = \sum_{i=0}^{n} q_i x^i$ and $r(x) = \sum_{i=0}^{n} r_i x^i$.

$\quad$ Let $qc = [q_0, q_1, ..., q_n]$
$\quad$ Let $rc = [r_0, r_1, ..., r_n]$
$\quad$ $qf = FFT(qc)$
$\quad$ $rf = FFT(rc)$
$\quad$ for $i = 0$ to n
$\quad\quad\quad pf_i = qf_i \cdot rf_i$
$\quad$ $pc = invFFT(pf)$

pc contains the desired coefficients of p.

(b) (4 points) Compute the FFT of the vector $[0, 1, 2, 3]$.

4. Logic Puzzle. **20 points.**

You are given a box and a collection of N cards as indicated in the following figures. Each card is gray on one side and white on the other side. Because of the the shapes of the cards, which exactly match the shape of the hole in the box, each card will fit in the box in only two possible ways (gray-side-up or white-side-up). Each card contains $2R$ circles lined up in two columns and R rows, each of which may be punched out (so it is a hole) or not. Your puzzle is to see if you can place all the cards in the box (each one either gray-side-up or white-side-up) so as to completely cover the bottom of the box, i.e. each of the $2R$ circle positions is covered by at least one card that has no hole punched out there.

For example, in the cards for Puzzle 1 (left example below), with 2 cards each with 2 rows, it is not possible to put them in the box to cover each circle, but with Puzzle 2 (right example below), with 2 cards each with 3 rows, it is possible. (Punched-out-circles are shown in black).

Cards for Puzzle 1

Cards for Puzzle 2
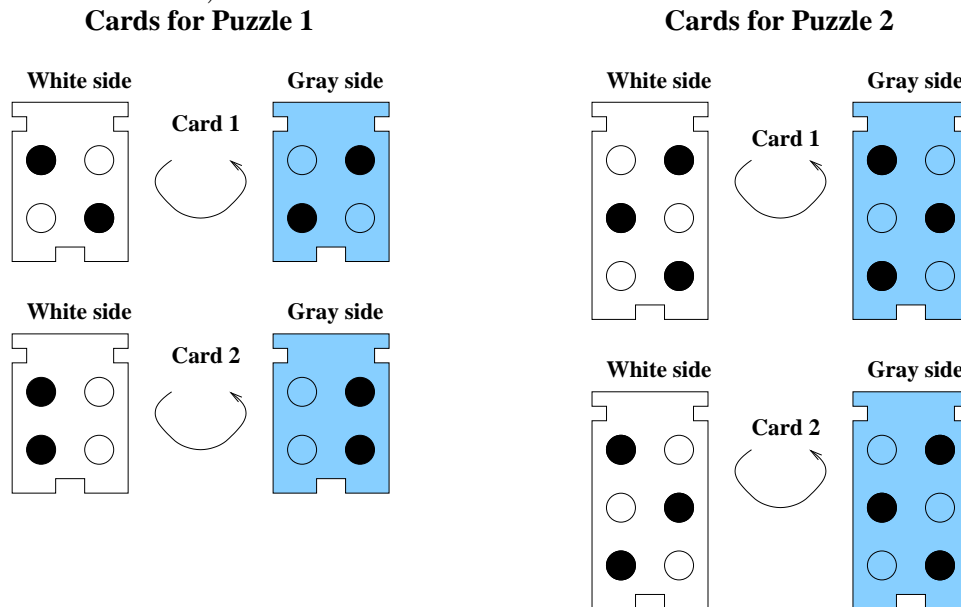
Your job is to show that solving this puzzle (deciding if the cards can be put in the box to cover each circle position) is NP-complete.

Hint: Show that 3-SAT can be reduced to an instance of solving this puzzle. Recall that 3-SAT is the problem of deciding whether a Boolean expression like the following can be satisfied (made True) by assigning True and False values to the variables x_i:

$$(x_1 \vee \neg x_4 \vee x_9) \wedge (\neg x_2 \vee \neg x_1 \vee x_3) \wedge \cdots \wedge (\neg x_5 \vee x_6 \vee x_7)$$

Try representing each Boolean variable in the 3-SAT problem by a card, and each term (like $(x_1 \vee \neg x_4 \vee x_9)$) by a row in each card.

.

5. Collinearity. **15 points.**

   Given n points in the plane, it is easy to determine if any three of them lie in a single straight line, in $O(n^3)$ time: for each set of three points, use the cross product test to see if they are collinear.

   Give a more efficient algorithm for this problem, one that runs in $o(n^3)$ time.

6. Dynamic Programming. **30 points.**

You have decided to make a CD of your favorite songs. You have already picked the songs, but you still need to decide in what order they should appear on the CD. Your task is the compute the "best" possible order for them to appear.

For each pair of songs a and b you've chosen a "compatibility rating," $compat[a, b]$, which is a positive number indicating how good you think b sounds when it's played right after a. (Note that $compat[a, b]$ and $compat[b, a]$ are generally *not* equal). Your job is to use dynamic programming to find an ordering of the songs which maximizes the sum of the compatibilities of consecutive songs. *You do not need to print out the actual ordering, just the maximum total compatibility.*

Let your N favorite songs have identification numbers from 1 to N. Your algorithm should run in $O(N^2 2^N)$ time.

(a) **(5 points)** List the table(s) your algorithm will use, and explain the meaning of each entry.

(b) **(10 points)** Specify the recursion and the base case(s) used by your algorithm.

(c) **(10 points)** Implement your algorithm in pseudocode. (Be sure to return the answer at the end.)

(d) **(5 points)** Analyze the running time of your algorithm; justify your answer briefly.

7. Short Answer **(25 points, 5 points each.)**

   Answer in the space allotted. For full credit justify your answer briefly.

   (a) In a disjoint set data structure with path compression and union-by-rank, when can the rank of a representative element differ from the depth of the tree representing its set?

   (b) Suppose we use optimized Huffman coding to compress a 50,000-symbol file containing 5 different symbols, each of which occurs equally often. What will the output file size be (excluding any header information), measured in the number of bits?

   (c) Consider the sets of problems P, NP, NP-Complete, and NP-Hard. List all known (proven) containment relationships between pairs of sets, i.e. all relationships of the form $A \subseteq B$. (don't bother with trivial ones like $A \subseteq A$).

   (d) Consider using RSA encryption with $n = 143 = 11 \cdot 13$. What is wrong with using $e = 3$ as the encryption key? What is the smallest e that would work?

   (e) Give a simple asymptotic expression (i.e. $\Theta(\cdot)$) for $T(n) = 2T(n/2) + n \log n$, $T(1) = 1$, n a power of 2.