1 [6 p]

Given $A = \{a \in \mathbb{N}^+ : a \leq 6\}$, with $\mathbb{N}^+$ as always the natural numbers starting at 1, let us define the following sets (with $a|b$ iff $\exists k(k \in \mathbb{N}^+ \wedge ka = b)$):

$$B = \left\{\frac{a}{b} : a, b \in A\right\}$$
$$C = \left\{\frac{a}{b} : a, b \in A \wedge b|a\right\}$$
$$D = \left\{\frac{a}{b} : a, b \in A \wedge a|b\right\}$$

Give the number of elements in these sets as follows:

1. [2 p] $\#(B) =$ 23
2. [2 p] $\#(C) =$ 6
3. [2 p] $\#(D) =$ 6

Note that the question is about the *cardinality* of sets, so the answers are numbers.

2 [7 p]

Given $A = \{a, b, c, d, e, f\}$, what is

1. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 0\}) = 1$
2. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 1\}) = 6$
3. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 2\}) = 15$
4. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 3\}) = 20$
5. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 4\}) = 15$
6. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 5\}) = 6$
7. [1 p] $\#(\{s \in \mathcal{P}(A) : \#(s) = 6\}) = 1$

3 [8 p]

With $A = \{n \in \mathbb{N}^+ : n \leq 20\}$ and $R = \{(a, b) \in A^2 : a|b\}$ compute the following images of R:

1. [2 p] $R(6) = \{6, 12, 18\}$
2. [2 p] $R(7) = \{7, 14\}$
3. [2 p] $R(2) = \{2, 4, 6, 8, 10, 12, 14, 16, 18, 20\}$
4. [2 p] $R(\{2, 5\}) = \{2, 4, 5, 6, 8, 10, 12, 14, 15, 16, 18, 20\}$

4 [19 p]

With $A = \{2, 3, 4, 5, 6, 7\}$, $R = \{(a, b) \in A^2 : a > b\}$, and $S = \{(a, b) \in A^2 : a|b\}$. We are looking at the composition $S \circ R$ in this task.

1. [3 p] $\#(S \circ R) = 21$
2. [3 p] $S \circ R(2) = \emptyset$
3. [3 p] $S \circ R(3) = \{2, 4, 6\}$
4. [3 p] $S \circ R(4) = \{2, 3, 4, 6\}$
5. [3 p] $S \circ R(7) = \{2, 3, 4, 5, 6\}$
6. [4 p] $S \circ R$ is ... (circle those that apply)

reflexive symmetric transitive antisymmetric

Many solutions to 2-5 produced sets of tuples, rather than numbers. If you produced one of those, you might want to review the definition of an *image* of a relation.

I marked those answers by making a deduction in the first (otherwise correct) answer, and marking the others by only regarding the right side of the tuples listed.

5 [12 p]

Assume you have an injection $j : A \hookrightarrow B$ and a surjection $s : B \twoheadrightarrow C$.

1. [1 p] Is their composition $s \circ j$ always injective?

   YES NO

2. [5 p] If yes, prove that it is. If no, show a counterexample. (A counterexample involves making the three sets A, B, and C concrete, giving two functions for j and s with the required properties, and showing how their composition is not injective.)

   A = {a, b}, B = {c, d}, C = {e}

   j = {(a, c), (b, d)}, s = {(c, e), (d, e)}

   s*j = {(a, e), (b, e)}

3. [1 p] Is their composition $s \circ j$ always surjective?

   YES NO

4. [5 p] If yes, prove that it is. If no, show a counterexample. (A counterexample involves making the three sets A, B, and C concrete, giving two functions for j and s with the required properties, and showing how their composition is not surjective.)

   A = {a}, B = {c, d}, C = {e, f}

   j = {(a, c)}, s = {(c, e), (d, f)}

   s*j = {(a, e)}

Some counterexamples that people constructed involved s and j that were not, in fact, *functions*. Some folks tried to prove that the composition was injective or surjective, and, obviously, failed, since it isn't.

A lesson here could be that if you are trying to prove something and find you can't, do consider the possibility that what you are trying to show just isn't so. Not always (see 8.4/5), but sometimes.

6 [11 p]

Consider the lower-case alphabet $A = \{"a", ..., "z"\}$ and the set
$C = A \cup \{"(", ")", "\neg", " \vee ", " \wedge "\}$ of characters.
We define a small language $\mathcal{L} \subseteq C^*$ of propositional formulae over the set of variable names $V = A^* \setminus \{\varepsilon\}$, and the following set of rules $R = \{R_1, R_2, R_3\}$ with

$$R_1 = \{(s, "\neg" s) : s \in C^*\}$$
$$R_2 = \{(s_1, s_2, "(" s_1 " \vee " s_2 ")") : s_1, s_2 \in C^*\}$$
$$R_3 = \{(s_1, s_2, "(" s_1 " \wedge " s_2 ")") : s_1, s_2 \in C^*\}$$

such that $\mathcal{L} = R[V]$.

1. [1 p] Show that $\mathcal{L} \subset C^*$ by giving a string $s \in C^*$ such that $s \notin \mathcal{L}$:

   $s = ($

   Hint: Make sure the strings are in $C^* \setminus \mathcal{L}$!

2. [3 p] Give three strings $s_1, s_2, s_3 \in C^* \setminus \mathcal{L}$ such that $(s_1, s_2, s_3) \in R_3$:

   $s_1 =)$
   $s_2 = ($
   $s_3 = () \wedge ()$ Note that $(s_1, s_2, s_3) \in R_3$ -- many solutions missed that point.

3. [7 p] Assume a function $E : V \longrightarrow \{0, 1\}$ that assigns every variable name a value in $\{0, 1\}$. Using **structural recursion**, define an evaluation function $\text{eval}_E : \mathcal{L} \longrightarrow \{0, 1\}$ that interprets the formulae in $\mathcal{L}$ in a way consistent with the usual interpretation of the symbols $\neg$, $\vee$, and $\wedge$ in propositional logic. **Use arithmetic operators (+, -, *, min, max) to compute with the values 0 and 1.**

$$\text{eval}_E : s \mapsto \begin{cases} E(s) & \text{for } s \in V \\ 1 - eval_E(s') & \text{for } s = \neg s' \\ max(eval_E(s_1), eval_E(s_2)) & \text{for } s = (s_1 \vee s_2) \\ min(eval_E(s_1), eval_E(s_2)) & \text{for } s = (s_1 \wedge s_2) \end{cases}$$

In many solutions the eval function could produce values outside of {0, 1} --- for example, if the second clause is simply -eval(s'), it may result in -1, and if the third is eval(s1) + eval(s2), you may get 2 etc.

7 [12 p]

Suppose we have a set A that is **totally ordered** by a relation $<$ on A. A function $f : A \longrightarrow A$ is called *strictly monotonic* iff for any $a, b \in A$ it is the case that $a < b \rightarrow f(a) < f(b)$.

1. [1 p] If a function f is strictly monotonic, does that imply it is **injective**? (circle answer)

   YES NO

2. [5 p] If yes, prove it. If no, provide a counterexample.
   (If you use a counterexample, you may use any totally ordered set that you find convenient, such as the natural/integer/rational/real numbers under the usual arithmetic order.)

   It is to show that $x \neq y$ implies $f(x) \neq f(y)$
   Since $x \neq y$, and A is totally ordered, then either $x < y$ or $y < x$.
   If the former, then $f(x) < f(y)$ and therefore $f(x) \neq f(y)$.
   If the latter, correspondingly.

   Some observed that x<y implies x!=y, which is correct, but in itself not helpful, because you need to show that x!=y implies f(x)!=f(y), so x!=y is the starting point of the proof. To get from there to x<y (or y<x), you need to invoke the *total* order property of <.
   One solution actually interpreted < as non-strict, and thus concluded that a "strictly monotonic" function in the above definition would not be injective. That conclusion is correct, and since I did not explicitly say that < is supposed to be strict, I gave it full marks. (Even if the term "strictly monotonic" and the use of the symbol < might have suggested the intent, the ambiguity ultimately was mine.)

3. [1 p] If a function f is strictly monotonic, does that imply it is **surjective**? (circle answer)

   YES NO

4. [5 p] If yes, prove it. If no, provide a counterexample.
   (If you use a counterexample, you may use any totally ordered set that you find convenient, such as the natural/integer/rational/real numbers under the usual arithmetic order.)

   $f : \mathbb{R} \longrightarrow \mathbb{R}$
   $x \mapsto 2^x$

   f is strictly monotonic, but not surjective.

8 [21 p]

Let us use $\mathbb{P}$ as the name for the set of all prime numbers, that is positive integers greater than 1 that are only divisible by 1 and themselves, so $\mathbb{P} = \{2, 3, 5, 7, 11, 13, ...\}$. You can use $\mathbb{P}$ in answering the following questions, and also the "divides" relation, defined as $a|b$ iff $\exists k(k \in \mathbb{N}^+ \wedge ka = b)$.

1. [3 p] Give a definition of the set F_n of all prime factors of a positive natural number $n \in \mathbb{N}^+$, i.e. all prime numbers that are divisors of n.

   $F_n = \{p \in \mathbb{P} : p|n\}$

2. [6 p] The number *n primorial* is the product of all prime numbers less than or equal to n, i.e. $\prod\{p \in \mathbb{P} : p \leq n\}$. Let us call the function that computes n primorial P, so for example, $P(3) = 2 \cdot 3 = 6$, $P(4) = 2 \cdot 3 = 6$, $P(5) = 2 \cdot 3 \cdot 5 = 30$, $P(6) = 2 \cdot 3 \cdot 5 = 30$, $P(7) = 2 \cdot 3 \cdot 5 \cdot 7 = 210$ and so forth. The first primorial number is defined to be $P(1) = 1$.
   Using **simple recursion**, give a definition of the function $P : \mathbb{N}^+ \longrightarrow \mathbb{N}^+$ computing n primorial for any $n \in \mathbb{N}^+$, as follows:

$$P : n \mapsto \begin{cases} 1 & \text{for } n = 1 \\ P(n-1) & \text{for } n > 1, n \notin \mathbb{P} \\ nP(n-1) & \text{for } n > 1, n \in \mathbb{P} \end{cases}$$

3. [2 p] Is the function $P : \mathbb{N}^+ \longrightarrow \mathbb{N}^+$... (circle the answer)

   (a) injective? YES (NO)

   (b) surjective? YES (NO)

4. [4 p] Using **simple recursion**, define an **injective function** $Q : \mathbb{N}^+ \hookrightarrow \mathbb{P}$.
Use the fact that for any $k \in \mathbb{N}^+$, the number $P(k) + 1$ is a prime number (a so-called *primorial prime*).
Hint: It's NOT as simple as mapping n to $P(n) + 1$. (Make sure you understand why that is)
The reason is that P is not injective, e.g. P(3) = P(4).

$$Q : n \mapsto \begin{cases} P(1) + 1 & \text{for } n = 1 \\ P(Q(n-1)) + 1 & \text{for } n > 1 \end{cases}$$

Many people answered P(n^2) + 1. (Of course that does not result in a recursive definition, but let's forget about this for a moment.) This is based on the realization that just P(n)+1 is not injective because not every n is a new prime, and so in a lot of cases P(n)+1 = P(n+1)+1, making Q not injective. This gave rise to the suspicion/hope that between any n^2 and the next (n+1)^2 there will be at least one prime so that the next Q(n+1) is different from Q(n), making Q injective.

Interestingly, that suspicion has a name, it's called *Legendre's conjecture*, and while it has been around for over a hundred years and is almost certainly true, it hasn't been proven yet. I gave full marks for that answer even if it does not use simple recursion, because it does ("almost certainly") produce an injective Q. However, actually *proving* it injective would involve proving Legendre's conjecture, which, sadly, none of the solutions did (you would have heard about it in the news otherwise).

5. [6 p] Prove that Q above is injective.
You may use the fact that $n \leq P(n)$ for all $n \in \mathbb{N}^+$ without needing to prove it.
Hint: Answering this might become easier if you use a result from a previous task.

Using the result from 7, it is sufficient to show that Q is strictly monotonic, i.e.
$a < b \to Q(a) < Q(b)$, or alternatively $Q(a) < Q(a + k)$ for $k \geq 1$.

For $k = 1$ we have $Q(a+1) \overset{(1)}{=} P(Q(a)) + 1 > P(Q(a)) \overset{(2)}{\geq} Q(a)$.

Assuming that $Q(a) < Q(a + k)$, we need to show that $Q(a) < Q(a + k + 1)$.

$Q(a+k+1) \overset{(1)}{=} P(Q(a+k)) + 1 > P(Q(a+k)) \overset{(2)}{\geq} Q(a+k) \overset{(3)}{>} Q(a)$. QED

(1) definition of Q(n) for n>1 (a+1 and a+k+1 are both >1, because a and k are > 0)
(2) using $n \leq P(n)$
(3) induction hypothesis

Most solutions skipped the induction, and contended themselves with showing that for any n, Q(n) < Q(n+1), waving in the general direction of the transitivity of < to imply monotonicity. I let that one slide and gave it full marks.

9 [7 p]

Let $A = \{a, b, c\}$ and $X = \{x, y\}$, and correspondingly A^* and X^* be the sets of finite sequences in A and X, respectively.

Using recursion over the structure of the sequence, define two **injections** $f : X^* \hookrightarrow A^*$ and $g : A^* \hookrightarrow X^*$, as follows. In both definitions, the first case deals with the empty sequence, the other cases "peel off" the first element in the sequence and the rest of the sequence is called s'.

$$f : s \mapsto \begin{cases} \varepsilon & \text{for } s = \varepsilon \\ af(s') & \text{for } s = xs', s' \in X^* \\ bf(s') & \text{for } s = ys', s' \in X^* \end{cases}$$

$$g : s \mapsto \begin{cases} \varepsilon & \text{for } s = \varepsilon \\ xxg(s') & \text{for } s = as', s' \in A^* \\ xyg(s') & \text{for } s = bs', s' \in A^* \\ yyg(s') & \text{for } s = cs', s' \in A^* \end{cases}$$

There are, of course, many ways of answering here. The important point is that the resulting f and g be injective, and also (which many answers got wrong) that they map to A* and X*, respectively.

10 [4 p]

1. [2 p] Is the composition $f \circ g$ of the two functions defined in the previous task... (circle all that apply)

   (injective) surjective

2. [2 p] Is the composition $g \circ f$ of the two functions defined in the previous task... (circle all that apply)

   (injective) surjective

11 [9 p]

Identify free and bound occurrences of variables in the following formula. Put a dot **above** a free variable occurrence, and **below** a bound one.
Note that variable symbols immediately following quantifiers do not count as "occurrences".

free

$$Py \vee \exists z(Qxzy) \rightarrow \forall y(Rxy \leftrightarrow \exists y(Py \rightarrow \forall x(Rzx)))$$

bound

12 [15 p]

Find a DNF for each of the following formulae

1. [5 p] $\neg((r \vee q) \leftrightarrow (q \vee p))$

   $(\neg p \wedge \neg q \wedge r) \vee (p \wedge \neg q \wedge \neg r)$

2. [5 p] $\neg((p \rightarrow q) \vee (q \rightarrow r) \vee (r \rightarrow p))$

   none

3. [5 p] $(p \rightarrow q) \wedge (q \rightarrow r) \wedge (r \rightarrow p)$

   $(p \wedge q \wedge r) \vee (\neg p \wedge \neg q \wedge \neg r)$

   Quite a few solutions produced formulae that were not, in fact, in DNF.

1 [6 p]

Given $A = \{4, 5, 6, 7, 8, 9\}$, suppose we define the following sets
(see the last page for the $\perp$ operator):

$$B = \left\{ \frac{a-b}{a+b} : a, b \in A \right\}$$

$$C = \left\{ \frac{a}{b} : a, b \in A \wedge a \perp b \right\}$$

Give the number of elements in these sets as follows:

1. [3 p] $\#(B) =$ 29
2. [3 p] $\#(C) =$ 22

Note that the question is about the *cardinality* of sets, so the answers are numbers.

2 [8 p]

With $A = \{n \in \mathbb{N}^+ : n \leq 20\}$ and $R = \{(a, b) \in A^2 : a \perp b\}$ compute the following images of R:

1. [2 p] $R(6) =$ {1, 5, 7, 11, 13, 17, 19}
2. [2 p] $R(7) =$ {1, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20}
3. [2 p] $R(2) =$ {1, 3, 5, 7, 9, 11, 13, 15, 17, 19}
4. [2 p] $R(\{2, 5\}) =$ {1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19}

3 [23 p]

With $A = \{2, 3, 4, 5, 6, 7\}$, $R = \{(a, b) \in A^2 : a \perp b\}$, and $S = \{(a, b) \in A^2 : a|b\}$. We are looking at the composition $S \circ R$ in this task.

1. [3 p] $\#(S \circ R) =$ 25
2. [3 p] $S \circ R(2) =$ {3, 5, 6, 7}
3. [3 p] $S \circ R(3) =$ {2, 4, 5, 6, 7}
4. [3 p] $S \circ R(6) =$ {5, 7}
5. [3 p] $S \circ R(7) =$ {2, 3, 4, 5, 6}
6. [4 p] $S \circ R$ is ... (circle those that apply)

… reflexive	TRUE	(FALSE)
… symmetric	TRUE	(FALSE)
… transitive	TRUE	(FALSE)
… antisymmetric	TRUE	(FALSE)

7. [4 p] R is ... (circle those that apply)

… reflexive	TRUE	(FALSE)
… symmetric	(TRUE)	FALSE
… transitive	TRUE	(FALSE)
… antisymmetric	TRUE	(FALSE)

4 [12 p]

Assume you have a surjection $s : A \twoheadrightarrow B$ and an injection $j : B \hookrightarrow C$.

1. [1 p] Is their composition $j \circ s$ always injective?

   YES NO

2. [5 p] If yes, prove that it is. If no, show a counterexample. (A counterexample involves making the three sets A, B, and C concrete, giving two functions for j and s with the required properties, and showing how their composition is not injective.)

   $A = C = \{a, c\}, B = \{b\}$
   $s = \{(a, b), (c, b)\}, j = \{(b, c)\}$
   $j \circ s = \{(a, c), (c, c)\}$

   The composition is not injective, since $j \circ s(a) = j \circ s(c) = c$.

3. [1 p] Is their composition $j \circ s$ always surjective?

   YES NO

4. [5 p] If yes, prove that it is. If no, show a counterexample. (A counterexample involves making the three sets A, B, and C concrete, giving two functions for j and s with the required properties, and showing how their composition is not surjective.)

   Same definitions as before. The composition is not surjective because $j \circ s(A) = \{c\} \neq C$.

One mistake that occurred a few times was giving definitions for j and s that were not, in fact, functions at all (either because they weren't unique mappings, or because they omitted values from the domain).

5 [8 p]

As we saw in the lecture on quantificational logic, $\exists x \forall y Rxy$ always implies $\forall x \exists y Rxy$ for any relation *R*. The converse, however, is not necessarily the case: $\forall x \exists y Rxy$ does not always mean that $\exists x \forall y Rxy$ is true.

1. [6 p] Define a binary relation R over a non-empty universe D (that you also need to define) such that $\forall x \exists y Rxy$ is true, and $\exists x \forall y Rxy$ is false.
   Hint: Keep in mind that the $\forall$ and $\exists$ operators are quantified over D.

   D = {a, b}

   R = {(a, a), (b, b)}

2. [2 p] Suppose $D = R = \emptyset$. What are the values of the formulae then?

$\forall x \exists y Rxy$	TRUE	FALSE
$\exists x \forall y Rxy$	TRUE	FALSE

Of course, there are many possible ways of constructing D and R. If they are empty, the first formula in the second part is true, because that is what happens when one universally quantifies over an empty set. Conversely, existentially quantifying over an empty set always yields false.

6 [17 p]

1. [2 p] Suppose we have a set A partially ordered by a strict order $<$. What would you need to show in order to demonstrate that $(A, <)$ is **not** well-founded? (in English/Swedish)

   An infinite descending chain in A. [or]
   A non-empty subset of A without a minimal element.

2. [8 p] The set $\mathcal{P}(\mathbb{Q})$ under strict set inclusion $\subset$ is not well-founded. Show this.

   $S = \left\{ \left[0, \frac{1}{n}\right] : n \in \mathbb{N}^+ \right\}$
   S has no minimal element.

   There are, of course, many ways of constructing an infinite descending chain. Including, incidentally, the one in the fourth part of this problem.

3. [1 p] Is the set $\mathcal{P}(\mathbb{N})$ under strict set inclusion $\subset$ well-founded? (circle answer)

   YES NO

4. [6 p] Prove it or provide a counterexample. (Hint: You can use the bijection between $\mathbb{N}$ and $\mathbb{Q}$ we discussed in the course without having to define it here.)

   $S = \{\{i : i \in \mathbb{N}^+, i > n\} : n \in \mathbb{N}^+\}$
   S has no minimal element.

   The bijection hint was meant for those who wouldn't find a simpler solution like the one above. If you found an answer to the second part of this problem, you already had an infinite descending chain in the rational numbers. With the bijection between those and the natural numbers, you could turn that into an infinite descending chain of sets of natural numbers "for free".

7 [10 p]

Suppose we have a directed graph (V, E). We want to define a function $r : V \longrightarrow \mathcal{P}(V)$ that computes for each vertex the set of vertices that one can reach from it by following zero or more directed edges.

We do this using a helper function $r' : V \times \mathcal{P}(V) \longrightarrow V$ that keeps track of the vertices we have visited already, so that we do not get stuck in cycles. Then r itself simply becomes

$$r(a) = r'(a, \emptyset)$$

The second argument of r' is the set of vertices we have visited already, initially empty.

Your task is to define r' using recursion:

$$r' : (a, S) \mapsto \begin{cases} S & \text{for } a \in S \\ S \cup \{a\} \cup \bigcup_{v \in E(a)} r'(v, S \cup \{a\}) & \text{otherwise} \end{cases}$$

Hint 1: Note that the edge relation E can be used to compute all the nodes that can be reached from a given vertex a in one hop: that set is simply the image of a under E, i.e. $E(a)$.

Hint 2: You might want to recall the notion of a “generalized union”, along with the associated notation.

Judging from the responses, this seemed a very difficult problem.

The first case handles the case where we run into a node we have visited already, i.e. at the end of a cycle. In that case, we return all the nodes we have seen so far.

In the second case we deal with a node we haven't encountered before. We return the union of two things: $S \cup \{a\}$, the set of all nodes we have seen so far (plus the current one), and $\bigcup_{v \in E(a)} r'(v, S \cup \{a\})$, the union of all those nodes we can reach from here along the edges going out from the current node (not forgetting to add the current node to the set of “visited nodes” we pass into r').

One might think that this second part would be sufficient, and that we would not need to explicitly add $S \cup \{a\}$ at the beginning, because all the calls to r' eventually run into the first case and will return the set S then. Except they don't: if a node has no outgoing edges, E(a) is the empty set, and no calls to r' will be made. That is why we need to include $S \cup \{a\}$. Alternatively, one could write r' to explicitly distinguish the case where the current node has outgoing arcs from when it doesn't, which would add another case to the two given in the problem.

8 [9 p]

Identify free and bound occurrences of variables in the following formula. Put a dot **above** a free variable occurrence, and **below** a bound one.
Note that variable symbols immediately following quantifiers do not count as "occurrences".

free

$$(\exists y(Py \vee Qxzy)) \rightarrow \forall x(Rxy \leftrightarrow \exists y(Pz \rightarrow \forall x(Rzx)))$$

bound

9 [15 p]

Find a DNF for each of the following formulae. Write “none” if a formula has no DNF.

1. [5 p] $(r \vee q) \rightarrow (q \vee p))$

   $p \vee q \vee (\neg q \wedge \neg r)$

2. [5 p] $(p \rightarrow q) \rightarrow ((q \rightarrow r) \vee (r \rightarrow p))$

   $p \vee \neg q \vee r \vee \neg r \vee (p \wedge \neg q) = p \vee \neg q \vee r \vee \neg r = \text{true}$

3. [5 p] $(p \rightarrow (q \wedge r)) \vee (q \rightarrow (p \wedge r)) \vee (r \rightarrow (p \wedge q))$

   $\neg p \vee \neg q \vee \neg r \vee (p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$

   The last two DNFs evaluate to TRUE. That does not mean they don't exist, only that they aren't very interesting. For example, $p \vee \neg q \vee r \vee \neg r$ is a perfectly respectable DNF, even if it is obviously always TRUE.

   Either way, responses that said that there was no DNF were scored incorrect. One exception was a response that said there was no DNF because the formulae were TRUE. I gave points for that one.

   Also, any response that wasn't a DNF was marked incorrect.

1 [12 p] avrg: 9.9 (83%), sd: 3.2

Suppose $A = \{n \in \mathbb{N} : 1 \leq n \leq 10\}$. For any non-empty set S of numbers, $\max S$ and $\min S$ are the largest and smallest numbers S, respectively.

1. [3p] $\#\{(a,b) \in A \times A : a \leq b\} =$ 55

2. [4p] $\max\left\{\frac{a}{b} : a,b \in A\right\} - \min\left\{\frac{a}{b} : a,b \in A\right\} =$ 99/10

3. [5p] $\min\left\{\frac{a}{b} : a,b \in A, a > b\right\} - \max\left\{\frac{a}{b} : a,b \in A, a \leq b\right\} =$ 1/9

2 [20 p] avrg: 13.0 (65%), sd: 3.5

Suppose $A = \{n \in \mathbb{N} : 1 \leq n \leq 10\}$ as before and a family of relations
$R_i = \{(a,b) \in A \times A : \mod(b,a) = i\}$ for any $i \in \mathbb{N}$, with $\mod(b,a)$ the remainder when dividing positive integer b by positive integer a. So, for example,
$R_3 = \{(a,b) \in A \times A : \mod(b,a) = 3\}$.

1. [2p] $\#R_4 =$ 8

2. [2p] $\#R_5 =$ 5

3. [2p] $\#R_0 =$ 27

4. [2p] $\#R_{10} =$ 0

5. [3p] $R_3(7) =$ {3, 10}

6. [3p] $R_1(2) =$ {1, 3, 5, 7, 9}

7. [3p] $R_1(A) =$ {1, 3, 4, 5, 6, 7, 8, 9, 10}

8. [3p] $R_3(A) =$ {3, 7, 8, 9, 10}

   Hint: Please review the definition of “image”, and be sure of what the kind of result is expected here.

The most frequent question I get about these kinds of tasks is whether there is a “shortcut”. In this case, one way to go about answering this question is to construct a little table representing the modulo operation for numbers between 1 and 10. Note that the table contains a number of regularities, and it can be constructed very easily. Just follow the first few rows to see how this is built.

		b									
	b mod a	1	2	3	4	5	6	7	8	9	10
a	1	0	0	0	0	0	0	0	0	0	0
	2	1	0	1	0	1	0	1	0	1	0
	3	1	2	0	1	2	0	1	2	0	1
	4	1	2	3	0	1	2	3	0	1	2
	5	1	2	3	4	0	1	2	3	4	0
	6	1	2	3	4	5	0	1	2	3	4
	7	1	2	3	4	5	6	0	1	2	3
	8	1	2	3	4	5	6	7	0	1	2
	9	1	2	3	4	5	6	7	8	0	1
	10	1	2	3	4	5	6	7	8	9	0

Once you have this table, answering 2.1 – 2.8 becomes pretty straightforward.

3 [16 p] avrg: 7.6 (47%), sd: 4.8

Suppose, as previously, $A = \{n \in \mathbb{N} : 1 \leq n \leq 10\}$ and a relation
$R_3 = \{(a, b) \in A \times A : \mod (b, a) = 3\}$.

Let $T = R_3 \circ R_3^{-1}$.

Here, too, things become a lot simpler once you have explicitly constructed the extensions of the relations involved. So for reference:

$R_3 = \{(4,3), (4,7), (5,3), (5,8), (6,3), (6,9), (7,3), (7,10), (8,3), (9,3), (10,3)\}$

$R_3^{-1} = \{(3,4), (7,4), (3,5), (8,5), (3,6), (9,6), (3,7), (10,7), (3,8), (3,9), (3,10)\}$

$T = \{(3,3), (7,7), (8,8), (9,9), (10,10), (3,7), (3,8), (3,9), (3,10), (10,3), (9,3), (8,3), (7,3)\}$

1. [3 p] $\#T = 13$
2. [3 p] $T(1) = \{\}$
3. [3 p] $T(7) = \{3, 7\}$
4. [3 p] $T(A) = \{3, 7, 8, 9, 10\}$
5. [4 p] T is ... (circle those that apply)

… reflexive	TRUE	(FALSE)
… symmetric	(TRUE)	FALSE
… transitive	TRUE	(FALSE)
… antisymmetric	TRUE	(FALSE)

4 [18 p] avrg: 10.0 (57%), sd: 6.0

Suppose you have **injections** $f : A \hookrightarrow B$ and $g : A \hookrightarrow B$, as well as a **non-empty** set $S \subset A$ (note that S is a **proper** subset of A). Now let's define a function $h : A \longrightarrow B$ as follows:

$$h : x \mapsto \begin{cases} f(x) & \text{for } x \in S \\ g(x) & \text{for } x \notin S \end{cases}$$

This function is not, in general, injective.
Whether it is injective depends on the definitions of A, B, f, g, and S.

1. [5p] Give definitions for A, B, f, g, and S such that the h above is injective.

   A = {a, b}

   B = {x, y}

   f = {(a, x), (b,y)}

   g = {(a, x), (b,y)}

   S = {a}

2. [5p] Give definitions for A, B, f, g, and S such that the h above is **not** injective.

   A = {a, b}

   B = {x, y}

   f = {(a, x), (b,y)}

   g = {(a, y), (b,x)}

   S = {a}

3. [8p] Give a general formal criterion, depending only on A, B, f, g, and S (not necessarily all of them), that defines the condition under which h is injective. (Hint: Remember, f and g are already injective.)

   h is injective iff $f(S) \cap g(A \setminus S) = \emptyset$

Note: You are **not** supposed to reiterate the definition of injectivity for h, but rather give an expression involving at most A, B, f, g, and S (but **not** h) that is true if an only if they lead to an injective h.

Common mistakes on the first two subquestions included answers that defined f and g in such a way that they were not functions from A to B. Often, f was defined only on S and g on A\S. Occasionally, the image of A under f or g was not in B.

On the third subquestion, some answers missed that the criterion was supposed to be true *if and only if* (*iff*) h was injective. So, for example, if f(A) and g(A) are disjoint, then h will be injective, but that's not required: h will also be injective in cases where f(A) and g(A) aren't disjoint (for example, it could be that f=g, and in fact then h will always be injective, for any S), so that criterion is too strict.

5 [18 p] avrg: 5.6 (31%), sd: 6.3

Suppose we have a rooted tree (T, R) with nodes T, links $R \subseteq T \times T$, and root a as well as a labeling function $\lambda : T \longrightarrow \mathbb{N}$ assigning each node in the tree a natural number.

We want to define a function $L : T \longrightarrow \mathbb{N}$ that computes for each node $n \in T$ the lowest number a node in the subtree rooted at n is labeled with (that subtree includes n itself). If the subtree consists only of n, its label $\lambda(n)$ is the lowest number.

As before, for any non-empty set S of numbers, $\min S$ is the lowest number in that set.

1. [10p] Define L using well-founded recursion. (Hint: You may use cases if you like, but it is possible to define this function without an explicit "base case.")

   $$L : n \mapsto \min\{\lambda(n)\} \cup \{L(n') : nRn'\}$$

   Note that the nRn' takes care of the "termination": if there is no child, is means there is no n', and the second set in the union above will simply be empty.

2. [8p] Define a strict partial order $\prec$ on T such that the poset $(T, \prec)$ is well-founded and your definition of L performs well-founded recursion on that poset. For all $n, n' \in T$...

   $$n' \prec n \iff nR^*n'$$

   R^* is the transitive closure of R. Specifying *only* the link relation itself would not result in a poset, since it is not a partial order.

   There are other ways of answering that question, for example using the closure of {n} under R, i.e. R[{n}]: $n' \prec n \iff n' \in R[\{n\}]$

   Many answers tried to use L to define the order. However, that does not work since, for instance, the labeling could give the same number to every node (there is nothing that would require it not to), and so there would be no way to distinguish between nodes, which a strict order would have to.

   Also, it would not make much sense to do so, since the point of this order is to demonstrate the well-definedness of L, so using L to define it would be oddly circular. However, this in itself would not make the answer wrong (only useless for its intended purpose).

   No proof is required. It is sufficient that the strict partial order is well-founded and your definition of L conforms to it.

   Hint: Make sure the partial order you define actually is one, i.e. that it has all the properties required from a strict partial order, including, for example, transitivity.

6 [10 p] avrg: 10.0 (100%), sd: 0.0

Suppose we have a set of four five characters $C = \{"a", "b", "(", ")"\}$
$C = \{"a", "b", "(", ")", ",", "\}$, the set $S = \{"a", "b"\}$ consisting only of the letters a and b, and a relation $R = \{(s_1, s_2, "(" s_1 ", " s_2 ")") : s_1, s_2 \in C^*\}$.

As you can see, R is a 3-place relation, and we compute the image of some set of strings $X \subseteq C^*$ under R by applying the relation to all pairs of strings in X, i.e.
$R(X) = \{y : x_1, x_2 \in X, (x_1, x_2, y) \in R\}$.

Now let $R^n(X)$ be the set that results from computing the image of some set $X \subseteq C^*$ under R n times in a row, with $R^0(X) = X$, $R^1(X) = R(X)$, $R^2 = R(R(X))$ and so forth, so that $R^{n+1}(X) = R(R^n(X))$.

1. [1p] Give an element in $R^0(C)$: $R^0(S)$: a

2. [2p] Give an element in $R^1(C)$: $R^1(S)$: (a,b)

3. [2p] Give an element in $R^2(C)$: $R^2(S)$: ((a,b),(b,a))

4. [5p] Suppose $U = \bigcup_{n \in \mathbb{N}} R^n(C)$. $U = \bigcup_{n \in \mathbb{N}} R^n(S)$

   Give an element in $R[C] \setminus U$, $R[S] \setminus U$ or write "none" if no such element

   exists: (a,(a,b))

   Note:
   $R[C]$ $R[S]$ here is the closure of C S under the set of relations that only consists of the relation R.

Unfortunately, the version that was printed in the exam papers contained several errors (see the corrections above in red – the blue answers relate to the corrected questions). Some answers seem to have guessed the intention and corrected for it, others answered the question as asked, some did something in between. Either way, I saw no fair way to grade the answers, so I decided to give everybody full marks on this question, thereby effectively removing the question from consideration (and slightly raising the average score).

7 [10 p] avrg: 9.3 (93%), sd: 1.9

Identify free and bound occurrences of variables in the following formula. Put a dot **above** a free variable occurrence, and **below** a bound one.
Note that variable symbols immediately following quantifiers do not count as "occurrences".

free

$$((\forall z(Py \to Qzx)) \leftrightarrow Pz) \to (Qxz \wedge ((\exists x(Qxz)) \leftrightarrow \exists z(Pz \to Px)))$$

bound

8 [15 p] avrg: 7.9 (53%), sd: 5.7

Find a DNF for each of the following formulae. Write “none” if a formula has no DNF.

1. [5 p] $(r \vee \neg q) \leftrightarrow (p \wedge q)$

   $(\neg p \wedge q \wedge \neg r) \vee (p \wedge q \wedge r)$

```
(p q r)
(0 0 0)  -->  0
(0 0 1)  -->  0
(0 1 0)  -->  1
(0 1 1)  -->  0
(1 0 0)  -->  0
(1 0 1)  -->  0
(1 1 0)  -->  0
(1 1 1)  -->  1
```

2. [5 p] $(p \barwedge (q \barwedge (r \barwedge s)))$

$\neg p \vee (q \wedge \neg s) \vee (q \wedge \neg r)$

```
(p q r s)
(0 0 0 0)  -->  1
(0 0 0 1)  -->  1
(0 0 1 0)  -->  1
(0 0 1 1)  -->  1
(0 1 0 0)  -->  1
(0 1 0 1)  -->  1
(0 1 1 0)  -->  1
(0 1 1 1)  -->  1
(1 0 0 0)  -->  0
(1 0 0 1)  -->  0
(1 0 1 0)  -->  0
(1 0 1 1)  -->  0
(1 1 0 0)  -->  1
(1 1 0 1)  -->  1
(1 1 1 0)  -->  1
(1 1 1 1)  -->  0
```

3. [5 p] $(p \to q) \barwedge ((q \to r) \barwedge (r \to p))$

$(p \wedge r) \vee (p \wedge \neg q) \vee (\neg q \wedge \neg r)$

```
(p q r)
(0 0 0)  -->  1
(0 0 1)  -->  0
(0 1 0)  -->  0
(0 1 1)  -->  0
(1 0 0)  -->  1
(1 0 1)  -->  1
(1 1 0)  -->  0
(1 1 1)  -->  1
```

As always, the minimal requirement was that the answer be a DNF. In some cases, the answer was “close” (e.g., a negation too many/too few, or one basic conjunction missing), and I gave partial points for those.

Also, of course, there is more than one correct DNF for each of these. The answers typically depend on whether they were obtained by transforming the formula or through a truth table. Either way is fine, neither is better than the other.

1 [20 p]

For the following sets of numbers, specify the smallest and the largest numbers, write NONE if there is no smallest or largest number, or EMPTY (in one of the two columns) if the set is empty.

All intervals are supposed to be intervals in the real numbers, $\mathbb{R}$. Similarly, all relations and operators are on the real numbers, unless explicitly stated otherwise.

set	smallest element	largest element
$\{x \in \mathbb{Z} : x > 4 \wedge x < 2\}$	EMPTY	
$\{1, 2, 3, 4\}$	1	4
$]1, 4]$	NONE	4
$\bigcap_{i \in \mathbb{N}^+} \left[0, \frac{1}{i}\right] = \{0\}$	0	0
$\bigcap_{i \in \mathbb{N}^+} \left]0, \frac{1}{i}\right] = \emptyset$	EMPTY	
$\bigcap_{i \in \mathbb{N}^+} \left[\frac{-1}{i}, \frac{1}{i}\right] = \{0\}$	0	0
$\bigcap_{i \in \mathbb{N}^+} \left]\frac{-1}{i}, 1\right] = [0, 1]$	0	1
$\mathrm{sqrt}([0, 0.1]) = [0, \sqrt{0.1}]$	0	$\sqrt{0.1}$
$\mathrm{sqrt}[[0, 0.1]] = [0, 1[$	0	NONE
$\mathrm{sqrt}[]0, 0.1]] =]0, 1[$	NONE	NONE
$\bigcup_{i \in \mathbb{N}^+} \left]\frac{1}{i+1}, \frac{1}{i}\right[=]0, 1[$	NONE	NONE
$\mathrm{inv}(]0, 0.1]) = [10, +\infty[$	10	NONE
$\mathrm{inv}[]0, 0.1]] =]0, 0.1] \cup [10, +\infty[$	NONE	NONE

$\mathrm{sqrt} : \mathbb{R}^+ \longrightarrow \mathbb{R}^+$ is the positive square root function, i.e. for every non-negative real number a, $\mathrm{sqrt}(a)$ is the non-negative real number such that $a = \mathrm{sqrt}(a) \cdot \mathrm{sqrt}(a)$.

$\mathrm{inv} : \mathbb{R} \setminus \{0\} \longrightarrow \mathbb{R} \setminus \{0\}$ is the inversion function, defined by $\mathrm{inv} : r \mapsto \frac{1}{r}$.

When looking at intersections of an infinite number of sets, it is important to keep in mind that any value that is an element of the intersection must be an element of each and every one of those sets. Take for example $\bigcap_{i \in \mathbb{N}^+} \left[0, \frac{1}{i}\right]$. If that intersection contained any positive real number in addition to 0, it would mean that there is a real number $r > 0$, such that $r \in \left[0, \frac{1}{i}\right]$ for every $i \in \mathbb{N}^+$. It's easy to see that this cannot be the case: for any $r > 0$, there is some natural number $k > \frac{1}{r}$, which means that $\frac{1}{k} < r$, and so $r \notin \left[0, \frac{1}{k}\right]$, and so r cannot be in the intersection.

2 [6p]

Define two sets A and B, as well as a function $f : A \longrightarrow B$, such that f is **surjective** and **not injective**.

$A = \{a, b\}$

$B = \{x\}$

$f : a \mapsto x$

Of course, if you already found the answer to the next question, you could “reuse” it here by simply choosing to make A and B the same set.

The idea was to start with an easier question to get you to think about surjectivity and injectivity in a simpler setting first, before tackling the harder problem.

3 [10p]

Define **one** set A, as well as a function $f : A \longrightarrow A$, such that f is **surjective** and **not injective**.

$A = \mathbb{N}$

$$f : a \mapsto \begin{cases} 0 & \text{if } a = 0 \\ a - 1 & \text{otherwise} \end{cases}$$

The crux here is that A has to be infinite.

4 [4p]

Suppose there is a function $f : A \longrightarrow A$ which is surjective and **not** injective, like the one you were asked to define in the previous task. This question is about a property of A (the domain and codomain of f) that implies that such a function exists, and which is also implied by the existence of such a function. (You do not need to prove this here.)

A surjective and **not** injective function $f : A \longrightarrow A$ exists if and only if

A is infinite.

(this must be a statement about the set A, and cannot involve f)

If you want to get a deeper understanding of this point, try to prove it. You can do this in two steps: (1) You show that if A is infinite, a function exists on it that is surjective but not injective. This you can show by taking our definition of an infinite set (one that is equivalent to a proper subset of itself), and use that definition to construct such a function.

(2) Now you need to show that if such a function exists, then A is infinite. Being infinite means that A must be equinumerous to a proper subset of itself. So given a function that is surjective but not injective, you need to find a proper subset of A that is the same size as A.

If you find this confusing, have a look at the solution above, and try to figure out what a suitable proper subset of A would be, and how it is related to f.

5 [10p]

Recall that a *directed graph* (V, E) is defined as a finite set V of vertices and a relation $E \subseteq V \times V$ between them.

This question is about the properties of that relation. In the table below, make one mark in each row for the property in the left column, depending on whether all, some, or no relations defining a graph have that property. Put the mark in the corresponding ALL box, if **all relations** defining a graph have the corresponding property, the NONE box, if **no relation** has it, and the SOME box if at least one relation does, and at least one does not.

	ALL	SOME	NONE
reflexive over V		X	
transitive		X	
symmetric		X	
antisymmetric		X	
asymmetric		X	

Graphs in our definition make no special assumptions about the relations that define them, so any (finite) relation could be a graph.

6 [10p]

Recall that a *rooted tree* is a graph (T, R) such that, if the set T of nodes is not empty, then there is a node $a \in T$ (the root) such that for every $x \in T$ with $x \neq a$ there is exactly one path from a to x. Like V in the previous question, $R \in T \times T$ is a relation on the set of nodes. To make things simpler, for this question we only consider non-empty trees, that is $T \neq \emptyset$.

This question is about the properties of the relations defining trees. In the table below, make one mark in each row for the corresponding property in the left column, depending on whether all, some, or no relations defining a tree have that property. Put the mark in the corresponding ALL box, if **all relations** defining a tree have the corresponding property, the NONE box, if **no relation** has it, and the SOME box if at least one relation does, and at least one does not.

	ALL	SOME	NONE
reflexive over T			X
transitive		X	
symmetric		X	
antisymmetric	X		
asymmetric	X		

The situation is different for trees, which are much more specialized and constrained structures than graphs. Since we only consider non-empty trees, none of them can be reflexive. (If we allowed the empty tree, then its link-relation R would also be empty, which is reflexive over the empty set.)

But there are trees whose link relation is transitive, viz. all those of link height 0 or 1. (Make sure you understand why that is.) And there is a tree whose link relation is symmetric, namely the tree consisting of only a root, whose link relation is therefore empty, which is symmetric.

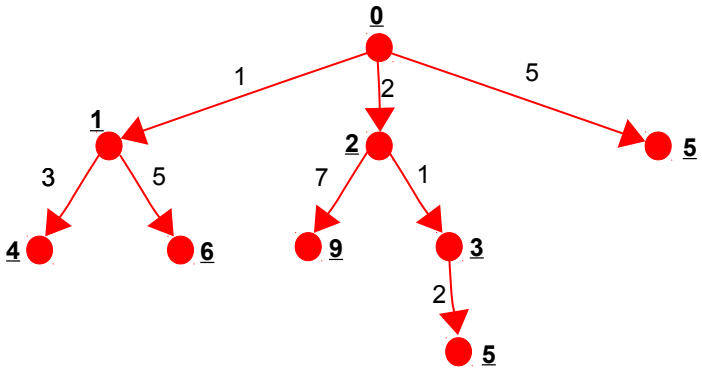
7 [16p]

Suppose we have a rooted tree (T, R) with nodes T, links $R \subseteq T \times T$.

We are also given a function $w : R \longrightarrow \mathbb{N}$ that assigns each *link* a natural number, let's call it the "weight" of that link.

1. [8p] Define a function $W : T \longrightarrow \mathbb{N}$ that maps each node in the tree to its "path weight", that is the sum of the weights of the links on the path from the root to that node.

   Example:



   The numbers next to the links are those assigned to them by the (given) function $w : R \longrightarrow \mathbb{N}$. The underlined numbers next to the nodes are those that your function $W : T \longrightarrow \mathbb{N}$ is supposed to compute in the case of this example.

$$W : n \mapsto \begin{cases} 0 & \text{if } R^{-1}(n) = \emptyset \\ w(m, n) + W(m) & \text{if } R^{-1}(n) = \{m\} \end{cases}$$

   Since we are working with a tree, the image of every node under the inverse link relation is either empty (if the node is the root) or a singleton set (its parent). Once you have those conditions, the recursive call "walks upward" in the tree, computing the path weight of the parent, and adds the weight of the link from the parent to this node. Of course, the path weight of the root is 0.

2. [4p] Formally (using math, not natural language) define the set $L \subseteq T$ of *leaf nodes* of a tree, i.e. nodes that do not have any children:

$$L = \{n \in T : R(n) = \emptyset\}$$

A node without children is one whose image under the link relation is the empty set. Note that this is the dual of the root, which is a node that has no parent, and whose image under the *inverse* link relation is empty.

3. [4p] Formally (again, using math, not natural language) define the set $M \subseteq L$ of leaf nodes in T with the smallest path weight (you may, indeed should, use L from the previous subtask):

$$M = \{n \in L : W(n) = \min\{W(m) : m \in L\}\}$$

Once you know the set of leaves, you can compute their smallest path weight as $\min\{W(m) : m \in L\}$, and then the set of all leaves with that path weight is, well, the set of all leaves whose path weight is that number.

8 [20 p]

Find a DNF for each of the following formulae. Write “none” if a formula has no DNF.

1. [5 p] $\neg((r \barwedge \neg q) \leftrightarrow (p \vee q))$

   $(\neg p \wedge \neg q \wedge \neg r) \vee (p \wedge \neg q \wedge r)$

```
(p q r)
(0 0 0)  -->  1
(0 0 1)  -->  0
(0 1 0)  -->  0
(0 1 1)  -->  0
(1 0 0)  -->  0
(1 0 1)  -->  1
(1 1 0)  -->  0
(1 1 1)  -->  0
```

2. [5 p] $\neg((p \barwedge q) \barwedge (r \barwedge s))$

   $(\neg q \wedge \neg s) \vee (\neg q \wedge \neg r) \vee (\neg p \wedge \neg s) \vee (\neg p \wedge \neg r)$

```
(p q r s)
(0 0 0 0)  -->  1
(0 0 0 1)  -->  1
(0 0 1 0)  -->  1
(0 0 1 1)  -->  0
(0 1 0 0)  -->  1
(0 1 0 1)  -->  1
(0 1 1 0)  -->  1
(0 1 1 1)  -->  0
(1 0 0 0)  -->  1
(1 0 0 1)  -->  1
(1 0 1 0)  -->  1
(1 0 1 1)  -->  0
(1 1 0 0)  -->  0
(1 1 0 1)  -->  0
(1 1 1 0)  -->  0
(1 1 1 1)  -->  0
```

3. [5 p] $\neg((p \mathbin{\overline{\wedge}} q) \rightarrow (q \mathbin{\overline{\wedge}} r))$

   $\neg p \wedge q \wedge r$

```
(p q r)
(0 0 0)  -->  0
(0 0 1)  -->  0
(0 1 0)  -->  0
(0 1 1)  -->  1
(1 0 0)  -->  0
(1 0 1)  -->  0
(1 1 0)  -->  0
(1 1 1)  -->  0
```

4. [5 p] $((p \rightarrow q) \mathbin{\overline{\wedge}} (q \rightarrow r)) \mathbin{\overline{\wedge}} (r \rightarrow p)$

   $(\neg p \wedge r) \vee (q \wedge r) \vee (\neg p \wedge \neg q)$

```
(p q r)
(0 0 0)  -->  1
(0 0 1)  -->  1
(0 1 0)  -->  0
(0 1 1)  -->  1
(1 0 0)  -->  0
(1 0 1)  -->  0
(1 1 0)  -->  0
(1 1 1)  -->  1
```

1 [10 p]

Suppose $A = \{n \in \mathbb{N} : 1 \leq n \leq 10\}$ and a family of relations on A,

$$R_i = \{(a, b) \in A \times A : \mod (ab, a + b) = i\}$$

for any $i \in \mathbb{N}$. Here, $\mod (m, n)$ is the remainder when dividing positive integer m by positive integer n, e.g. $\mod (14, 4) = 2$, and $\mod (5, 7) = 5$.

For example, $R_3 = \{(a, b) \in A \times A : \mod (ab, a + b) = 3\}$.

Compute the following cardinalities:

1. [2 p] $\#R_5 = 9$
2. [2 p] $\#R_6 = 8$
3. [2 p] $\#R_9 = 5$
4. [4 p] $\#(R_5 \circ R_6) = 4$

R_5: #{[4 3] [5 5] [3 4] [1 5] [2 7] [5 10] [7 2] [10 5] [5 1]}
R_6: #{[6 8] [4 7] [1 6] [2 8] [8 2] [6 1] [7 4] [8 6]}
R_9: #{[9 9] [6 9] [1 9] [9 1] [9 6]}
$R_5 \circ R_6$: #{[6 5] [8 7] [7 3] [4 2]}

2 [20 p]

Given a function $f : A \longrightarrow B$ and an injection $g : B \hookrightarrow C$. The sets A, B, and C are **not empty**.

1. [5 p] Is their composition $g \circ f : A \longrightarrow C$ injective (circle answer)?

   Never Sometimes Always

   (Circle "Never" if a function f and an injective function g could never be composed in this way to result in an injective function. Circle "Sometimes", if some such compositions would be injective, others would not be, depending on f and g. Circle "Always" if all such compositions would be injective.)

2. [15 p] If you answered "Never" or "Always" to the question above, prove it. If you answered "Sometimes", construct an example where $g \circ f$ is injective and one where it is not. (Constructing an example would mean defining sets A, B, and C, as well as the two functions, according to the decription above.)

   composition injective:

   $A = B = C = \{a\}$

   $f = g = \{(a, a)\}$

   composition not injective:

   $A = B = C = \{a, b\}$

   $f = \{(a, a), (b, a)\}$

   $g = \{(a, a), (b, b)\}$

3 [30 p]

Suppose you have a graph (V, E) with vertices V and edges $E \subseteq V \times V$. Also given is a function $w : E \longrightarrow \mathbb{N}^+$, assigning each edge a positive natural number as its *edge weight*.

In this task, we represent a *path* in the graph as a finite sequence of elements in V. In other words: a path in this graph is an element of V^*, such that each pair of successive vertices in it are connected by an edge. For example, suppose $p = abdcfbe$. If $a, b, c, d, e, f \in V$ then $p \in V^*$, and if $(a, b), (b, d), (d, c), (c, f), (f, b), (b, e) \in E$, then p is a path in our graph. Its *path weight* is the sum of the edge weights of the edges in it. Note that the same vertex can appear any number of times in a path. The same is true for the edges, and if an edge occurs multiple times in a path, its edge weight is counted for each occurrence. For example, if $p = abdab$, then we add the edge weights for (a, b), (b, d), (d, a), and (a, b) again to get the path weight for p.

We want to define a function

$$P : \mathbb{N} \longrightarrow \mathcal{P}(V^*)$$

which takes a natural number and computes a set of paths in the graph. Specifically, $P(n)$ is **the set of <u>all</u> paths** p in our graph such that

- p **visits every vertex at least once**, and
- its path weight is less than or equal to n.

In order to define P, we will use a helper function

$$P' : V \times V^* \times \mathcal{P}(V) \times \mathbb{N} \longrightarrow \mathcal{P}(V^*)$$

$P'(a, p, S, n)$ is the the set of all paths in our graph of the form paq, i.e. paths consisting of a path p, followed by vertex a, followed by path q, with the following properties:

- paq is a valid path in the graph,
- q visits all the vertices in S,
- the path weight of q is less than or equal to n.

Using P', we can define P as follows:

$$P : n \mapsto \bigcup_{a \in V} P'(a, \varepsilon, V \setminus \{a\}, n)$$

1. [25 p] Define P' recursively.

$$P' : (a, p, S, n) \mapsto \begin{cases} \{pa\} \cup \bigcup_{b \in \{v \in E(a) : w(a,v) \leq n\}} P'(b, pa, \emptyset, n - w(a, b)) & \text{for } S = \emptyset \\ \bigcup_{b \in \{v \in E(a) : w(a,v) \leq n\}} P'(b, pa, S \setminus \{b\}, n - w(a, b)) & \text{otherwise} \end{cases}$$

This one was of course the most difficult task. The following parts of the problem description were frequently misunderstood or ignored:

- The function is supposed to return a *set* of sequences, in fact the set of *all* sequences meeting the two following criteria.
- The first criterion was that every vertex occur at least once in a result sequence, but it can occur multiple times.
- The second criterion was that the path weight of the remaining result sequence be less than or equal to n.

Consequently, once you found a path that met those criteria, you needed to make it part of the result, but also to keep searching for longer paths, as long as the path weight had not exceeded n. That's what the first clause above does. The second clause handles the case that the paths so far have not yet touched all vertices, so you just keep on trucking.

Some students found a very elegant way of collapsing these two cases into one (note that once S is empty, so is $S \setminus \{b\}$):

$$P' : (a, p, S, n) \mapsto \{pa : S = \emptyset\} \cup \bigcup_{b \in \{v \in E(a) : w(a,v) \leq n\}} P'(b, pa, S \setminus \{b\}, n - w(a, b))$$

A few minor errors I let slide if the overall construction worked. For example, there seems to be a lot of confusion about the difference between the empty set $\emptyset$ and the empty sequence ε. Also, many people forgot to put a path they wanted to return into braces, i.e. to return a singleton set instead of just the path.

Another error which I also let pass if the rest worked, was to not actually test that the weight of the next edge $w(a, v)$ isn't greater than n under the union sign, i.e. when constructing the set of vertices to move to next before actually calling P'. Instead, in many solutions all vertices reachable from the current one were explored and the call just did $P'(..., n - w(a, b))$. This, of course, meant that the last argument (that is, n) could become negative, which had to be sorted out by adding a case in the definition of P' testing for that possibility. Technically, that's in violation of the definition of P', which demands a natural number as its last argument, but it's a reasonable choice if one were to implement this, so I let that one go.

2. [5 p] In order to ensure that P' terminates, we require a **well-founded strict order** $\prec$ of its arguments, such that for any (a, p, S, n) that P' is called on, it will only ever call itself on $(a', p', S', n') \prec (a, p, S, n)$. Define such an order:

$$(a', p', S', n') \prec (a, p, S, n) \iff n' < n$$

The important bit here is that, unlike the Hamiltonian path search, termination of this recursion is controlled by n. The parameter S is used to determine when paths are made part of the result (viz. when it is empty, i.e. all vertices have been visited at least once), but it has no effect on termination. It is true, of course, that it becomes monotonically smaller as the recursion progresses, but not *strictly* so: while it will never grow in the recursive call, it can very well stay the same, viz. when the next node has already occurred in the path so far, and so isn't contained in S. Consequently, that set cannot be used to construct a *strict* order.

4 [20 p]

Suppose we have a tree (T, R) and a function $\lambda : T \longrightarrow \mathbb{N}$ assigning each node in the tree a natural number as a label.

1. [5 p] Define the set $T_{\mathbb{P}} \subseteq T$ consisting of all nodes that are labeled with a number in $\mathbb{P}$, i.e. a prime number (you do not have to define prime numbers, just use $\mathbb{P}$ in your definition):

   $T_{\mathbb{P}} = \{n \in T : \lambda(n) \in \mathbb{P}\}$

2. [5 p] Define the set $A_{x,y} \subseteq T$ consisting of all nodes that have label x and that have at least one child with label y (for $x, y \in \mathbb{N}$):

   $A_{x,y} = \{n \in T : \lambda(n) = x \wedge \exists m \in R(n) : \lambda(m) = y\}$

   An elegant alternative that some students found is
   $A_{x,y} = \{n \in T : \lambda(n) = x \wedge y \in \lambda(R(n))\}$

3. [5 p] Define the set $B_{x,y} \subseteq T$ consisting of all nodes that have label x and have at least one "grandchild", i.e. one child of a child, with label y (for $x, y \in \mathbb{N}$):

   $B_{x,y} = \{n \in T : \lambda(n) = x \wedge \exists m \in R(R(n)) : \lambda(m) = y\}$

4. [5 p] Define the set $C_n \subseteq T$, consisting of all nodes that are descendents (children, children of children, children of children of children and so on) of node $n \in T$:

   $C_n = R^+(n)$

There are various ways of answering this. The one above uses the transitive closure of R (which our book writes as R^*). You could also use the closure of $R[\{n\}]$, as long as you remember that n itself is an element of it, and take it out, i.e. $C_n = R[\{n\}] \setminus \{n\}$. Some folks have used the C_m of other nodes, which is quite clever, or defined a function to help them compute C_n. Assuming they are technically correct, all of those are perfectly fine answers.

5 [20 p]

1. [5 p] $((p \to q) \barwedge (q \to r)) \wedge ((r \to s) \barwedge (s \to p))$

$(\neg p \wedge q \wedge \neg r \wedge s) \vee (p \wedge \neg q \wedge r \wedge \neg s)$

```
(p q r s)
(1 1 1 1)  -->  0
(1 1 1 0)  -->  0
(1 1 0 1)  -->  0
(1 1 0 0)  -->  0
(1 0 1 1)  -->  0
(1 0 1 0)  -->  1
(1 0 0 1)  -->  0
(1 0 0 0)  -->  0
(0 1 1 1)  -->  0
(0 1 1 0)  -->  0
(0 1 0 1)  -->  1
(0 1 0 0)  -->  0
(0 0 1 1)  -->  0
(0 0 1 0)  -->  0
(0 0 0 1)  -->  0
(0 0 0 0)  -->  0
```

2. [5 p] $\neg(((p \barwedge q) \to (q \barwedge r)) \to ((r \barwedge s) \to (s \barwedge p)))$

$(p \wedge \neg r \wedge s)$

```
(p q r s)
(1 1 1 1)  -->  0
(1 1 1 0)  -->  0
(1 1 0 1)  -->  1
(1 1 0 0)  -->  0
(1 0 1 1)  -->  0
(1 0 1 0)  -->  0
(1 0 0 1)  -->  1
(1 0 0 0)  -->  0
(0 1 1 1)  -->  0
(0 1 1 0)  -->  0
(0 1 0 1)  -->  0
(0 1 0 0)  -->  0
(0 0 1 1)  -->  0
(0 0 1 0)  -->  0
(0 0 0 1)  -->  0
(0 0 0 0)  -->  0
```

3. [5 p] $(p \leftrightarrow q) \wedge ((p \rightarrow r) \barwedge (q \rightarrow s))$

$(p \wedge q \wedge \neg r) \vee (p \wedge q \wedge \neg s)$

```
(p q r s)
(1 1 1 1)  -->  0
(1 1 1 0)  -->  1
(1 1 0 1)  -->  1
(1 1 0 0)  -->  1
(1 0 1 1)  -->  0
(1 0 1 0)  -->  0
(1 0 0 1)  -->  0
(1 0 0 0)  -->  0
(0 1 1 1)  -->  0
(0 1 1 0)  -->  0
(0 1 0 1)  -->  0
(0 1 0 0)  -->  0
(0 0 1 1)  -->  0
(0 0 1 0)  -->  0
(0 0 0 1)  -->  0
(0 0 0 0)  -->  0
```

4. [5 p] $(p \rightarrow q) \barwedge ((p \leftrightarrow r) \barwedge (q \leftrightarrow s))$

$(p \wedge \neg q) \vee (p \wedge q \wedge r \wedge s) \vee (\neg p \wedge q \wedge \neg r \wedge s) \vee (\neg p \wedge \neg q \wedge \neg r \wedge \neg s)$

```
(p q r s)
(1 1 1 1)  -->  1
(1 1 1 0)  -->  0
(1 1 0 1)  -->  0
(1 1 0 0)  -->  0
(1 0 1 1)  -->  1
(1 0 1 0)  -->  1
(1 0 0 1)  -->  1
(1 0 0 0)  -->  1
(0 1 1 1)  -->  0
(0 1 1 0)  -->  0
(0 1 0 1)  -->  1
(0 1 0 0)  -->  0
(0 0 1 1)  -->  0
(0 0 1 0)  -->  0
(0 0 0 1)  -->  0
(0 0 0 0)  -->  1
```

1 [20 p]

Suppose we have a graph (V, E) with vertices V and edges $E \subseteq V \times V$, as well as a labeling function $\lambda : V \longrightarrow \mathbb{N}$, assigning each vertex a natural number.

Recall that a *path* in this graph is a non-empty finite sequence $v_0 v_1 ... v_n \in V^*$, such that for an $i \in \{0, ..., n-1\}$ y we have $(v_i, v_{i+1}) \in E$. The number n, corresponding to the number of edges connecting the vertices in the path (and one less than the number of vertices in the sequence representing it), is called its *length*.

1. [10 p] Define a function $g : V \times \mathbb{N} \longrightarrow \mathcal{P}(V)$ such that for any vertex $v \in V$ and any natural number $k \in \mathbb{N}$, $g(v, k)$ is the set of all vertices $w \in V$ such that there is a path of length 1 or more from v to w, and that are labeled by λ with k.

$$g : V \times \mathbb{N} \longrightarrow \mathcal{P}(V)$$

$$v, k \mapsto \{w \in E^+(v) : \lambda(w) = k\}$$

Here, E^+ is the transitive closure of E. We also introduced the notation E^* for it in the lecture, so using that would have been fine, too.

2. [10 p] Define a function $h : V \longrightarrow \mathcal{P}(V)$ such that for any vertex $v \in V$ the value of $h(v)$ is the set of all vertices w such that there is a path of length 1 or more from v to w and a path of length 1 or more from w to v.

$$h : V \longrightarrow \mathcal{P}(V)$$

$$v \mapsto \{w \in E^+(v) : v \in E^+(w)\}$$

Hint: You do not need to use recursion in the above two answers (but it's okay if you use it, as long as the answer is correct).

2 [20 p]

Suppose you have an infinite set X and in **injection** $f : X \hookrightarrow \mathbb{N}$.

Note that this implies that X and $\mathbb{N}$ are equinumerous. (Make sure you understand why that is the case.)

The goal is to use f to define a **bijection** $g : \mathbb{N} \longleftrightarrow X$.

Note that we cannot simply invert f – while injectivity guarantees that every $n \in \mathbb{N}$ **is mapped to at most once** by f, some n may not be mapped to at all. For any $n \in \mathbb{N}$, either $f^{-1}(n) = \emptyset$, i.e. n was not mapped to by f, or $f^{-1}(n) = \{x\}$, the singleton set of the one value $x \in X$ mapped to n by f. Let us call the set of all values mapped to by f by the name M, i.e. $M = f(X)$.

Example: For instance, suppose $X = \{a, b, c\}^*$, i.e. the set of all finite strings of a, b, and c. f might then be $\{(aca, 2), (bbccaa, 5), (ccc, 7), (cbabc, 12), ...\}$ (listed in order of the number mapped to, so there is no mapping to 0, 1, 3, 4, 6, 8, 9 etc.). So in this case, M would be $\{2, 5, 7, 12, ...\}$.

We define the bijection g using a helper function $h : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$.
$h(n, k)$ is the $(n + 1)^{th}$ number in M (in the usual numerical order) greater or equal to k. Since all natural numbers are greater or equal to 0, $h(n, 0)$ is simply the $(n + 1)^{th}$ number in M, which we then can use to define the bijection g as follows:

$$g : \mathbb{N} \longleftrightarrow X$$
$$n \mapsto x \text{ with } f^{-1}(h(n, 0)) = \{x\}$$

To help you understand how to define h, note that, in the example, 7 is the $(2 + 1)^{th}$, i.e. third, number greater or equal to 0 in $M = \{2, 5, 7, 12, ...\}$, but it is also the $(1 + 1)^{th}$, i.e. second, number greater or equal to, for example, 3, and the $(0 + 1)^{th}$, i.e. first, number greater or equal to 6. Therefore, in the example, the following calls to h all yield the same result:

$$h(2, 0) = h(2, 1) = h(2, 2) = h(1, 3) = h(1, 4) = h(1, 5) = h(0, 6) = h(0, 7) = 7.$$

Understanding these equivalences should give you an idea how to construct the definition of h.

So, in the case of the example, $g(2)$ will call $h(2, 0)$, resulting in 7, and then $f^{-1}(7) = \{ccc\}$, and thus $g(2) = ccc$. Similarly, $g(1) = bbccaa$, $g(0) = aca$, etc.

Define h recursively.

$$h : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$$

$$n, k \mapsto \begin{cases} k & \text{if } f^{-1}(k) = \{x\} \wedge n = 0 \\ h(n-1, k+1) & \text{if } f^{-1}(k) = \{x\} \wedge n > 0 \\ h(n, k+1) & \text{if } f^{-1}(k) = \emptyset \end{cases}$$

Many answers used $k \in M$ instead of $f^{-1}(k) = \{x\}$, and correspondingly $k \notin M$ for $f^{-1}(k) = \emptyset$, which is fine.

3 [30 p]

Suppose you have a graph (V, E) with vertices V and edges $E \subseteq V \times V$.

As before, a *path* in this graph is a non-empty finite sequence $v_0 v_1 ... v_n \in V^*$, such that for an $i \in \{0, ..., n-1\}$ y we have $(v_i, v_{i+1}) \in E$. The number n, corresponding to the number of edges connecting the vertices in the path (and one less than the number of vertices in the sequence representing it), is called its *length*.

A *cycle* is a path of at least length 1 where the first and the last vertex are the same, so $v_0 = v_n$. A *simple cycle* is a cycle where every vertex occurs at most once, except for the first and last, which occurs exactly twice.

This task is about defining a function $C : V \longrightarrow \mathcal{P}(V^*)$ that for any vertex $v \in V$ computes **the set of all simple cycles** starting (and therefore also ending) at v.

We shall do so using a helper function $C' : V \times V^* \times V \longrightarrow \mathcal{P}(V^*)$, such that $C'(v, p, w)$ is the set of all simple cycles that (a) start (and end) at v, (b) then follow the path p, and (c) then continue with vertex w. In other words, $C'(v, p, w)$ is the set of all simple cycles that begin with vpw.

Using this, we can define C as follows (remember that ε represents the empty sequence):

$$
\begin{aligned}
&C : V \longrightarrow \mathcal{P}(V^*) \\
&\quad v \mapsto \bigcup_{w \in E(v)} C'(v, \varepsilon, w)
\end{aligned}
$$

Convince yourself that this results in all simple cycles starting at v if C' behaves as described above.

1. [20 p] Define C' recursively. You may find it useful to look at the *set* of all vertices occurring in a path $p \in V^*$. You can use the notation $set(p)$ for this purpose, i.e. if p is the path $v_0 v_1 ... v_n$, then $set(p)$ is the set $\{v_0, v_1, ..., v_n\}$.

   $C' : V \times V^* \times V \longrightarrow \mathcal{P}(V^*)$

$$v, p, w \mapsto \begin{cases} \{vpw\} & \text{if } v = w \\ \bigcup_{x \in E(w)} C'(v, pw, x) & \text{if } v \neq w \wedge w \notin set(p) \\ \emptyset & \text{if } v \neq w \wedge w \in set(p) \end{cases}$$

   One answer collapsed the cases into an elegant one-liner, roughly like this:

$$v, p, w \mapsto \{vpw : v = w\} \cup \bigcup_{x \in \{y \in E(w) : v \neq w \wedge w \notin set(p)\}} C'(v, pw, x)$$

2. [10 p] In order to ensure that C' terminates, we require a **well-founded strict order** $\prec$ of its arguments, such that for any (v, p, w) that C' is called on, it will only ever call itself on $(v', p', w') \prec (v, p, w)$. Define such an order:

   $(v', p', w') \prec (v, p, w) \iff set(p') \supset set(p)$

Note that the order must rely on the *set* of symbols in the partial path p. It is true, of course, that p' is always also a prefix of p, but using the prefix property to establish the order does not work because there are infinite chains in it (in other words: sequences can get longer forever, but there are only a finite number of vertices, so if we add a new one at every step, we will eventually terminate).

Hint: A correct answer to this question must have three properties.

1. It must be a strict order.
2. It must be well-founded, i.e. there cannot be an infinite descending chain in that order.
3. Your definition of C' must conform to it, i.e. any recursive call in it must be called on a smaller (according to the order) triple of arguments.

4 [20 p]

Suppose we have a graph (V, E), as usual with vertices V and edges $E \subseteq V \times V$, as well as a function $w : E \longrightarrow \mathbb{N}$ assigning each edge a natural number as a *weight*.

1. [5 p] Define the set $E_{\leq k} \subseteq E$ consisting of all edges in E with weight not more than k:

   $E_{\leq k} = \{e \in E : w(e) \leq k\}$

2. [5 p] Define the function $R : V \times \mathbb{N} \longrightarrow \mathcal{P}(V)$, such that $R(v, n)$ is the set of all vertices in V that can be reached from v in exactly n steps, and $R(v, 0) = \{v\}$.

   $R : V \times \mathbb{N} \longrightarrow \mathcal{P}(V)$

   $$v, n \mapsto \begin{cases} \{v\} & \text{if } n = 0 \\ E(R(v, n-1)) & \text{if } n > 0 \end{cases}$$

3. [5 p] Define the relation $P \subseteq V \times V$ such that for any two vertices $v, w \in V$, it is the case that $(v, w) \in P$ iff there is a path from v to w in the graph (V, E).

   $P = E^+$

4. [5 p] Define the relation $D \subseteq V \times V$ on the vertices in V such that for any two vertices $v, w \in V$ it is the case that $(v, w) \in D$ iff there is a path p from v to w and another path q from w to v that has the same length as p.

   $D = \{(v, w) \in V \times V : \exists n \in \mathbb{N}^+ : w \in R(v, n) \wedge v \in R(w, n)\}$

5 [10 p]

Find a DNF for each of the following formulae. Write “none” if a formula has no DNF.

1. [5 p] $((p \to q) \mathbin{\overline{\wedge}} (q \leftrightarrow r)) \wedge ((r \to s) \mathbin{\overline{\wedge}} (s \leftrightarrow p))$

$$
\begin{aligned}
&(p \wedge q \wedge \neg r \wedge \neg s) \\
\vee&(p \wedge \neg q \wedge r \wedge \neg s) \\
\vee&(p \wedge \neg q \wedge \neg r \wedge \neg s) \\
\vee&(\neg p \wedge q \wedge \neg r \wedge s) \\
\vee&(\neg p \wedge \neg q \wedge r \wedge s) \\
\vee&(\neg p \wedge \neg q \wedge r \wedge \neg s)
\end{aligned}
$$

```
(p q r s)
(1 1 1 1)  -->  0
(1 1 1 0)  -->  0
(1 1 0 1)  -->  0
(1 1 0 0)  -->  1
(1 0 1 1)  -->  0
(1 0 1 0)  -->  1
(1 0 0 1)  -->  0
(1 0 0 0)  -->  1
(0 1 1 1)  -->  0
(0 1 1 0)  -->  0
(0 1 0 1)  -->  1
(0 1 0 0)  -->  0
(0 0 1 1)  -->  1
(0 0 1 0)  -->  1
(0 0 0 1)  -->  0
(0 0 0 0)  -->  0
```

2. [5 p] $\neg(((p \barwedge q) \rightarrow (q \barwedge r)) \leftrightarrow ((r \barwedge s) \rightarrow (s \barwedge p)))$

$$
\begin{aligned}
&(p \wedge q \wedge \neg r \wedge s) \\
\vee&(p \wedge \neg q \wedge \neg r \wedge s) \\
\vee&(\neg p \wedge q \wedge r \wedge s) \\
\vee&(\neg p \wedge q \wedge r \wedge \neg s))
\end{aligned}
$$

```
(p q r s)
(1 1 1 1)  -->  0
(1 1 1 0)  -->  0
(1 1 0 1)  -->  1
(1 1 0 0)  -->  0
(1 0 1 1)  -->  0
(1 0 1 0)  -->  0
(1 0 0 1)  -->  1
(1 0 0 0)  -->  0
(0 1 1 1)  -->  1
(0 1 1 0)  -->  1
(0 1 0 1)  -->  0
(0 1 0 0)  -->  0
(0 0 1 1)  -->  0
(0 0 1 0)  -->  0
(0 0 0 1)  -->  0
(0 0 0 0)  -->  0
```

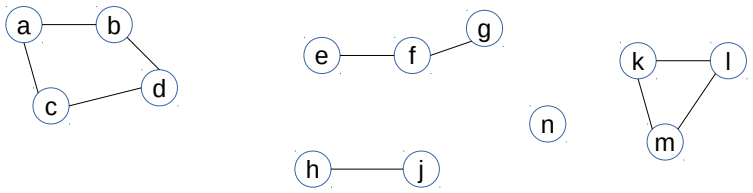


Figure 1: Example, villages and roads between them.

1 [10p]

Suppose we have a graph (V, R) with vertices V representing villages and edges $R \subseteq V \times V$ representing the roads between them, similar to the one in Figure 1 above. The road relation is *symmetric*, that is $\forall v_1, v_2 \in V \ ((v_1, v_2) \in R \Leftrightarrow (v_2, v_1) \in R)$.

1. [5 p] In the example above in Figure 1, we have the vertices
   $V = \{a, b, c, d, e, f, g, h, j, k, l, m, n\}$.
   Give the edge relation R for the example:

$$R = \{\ (a,b),(b,a),(a,c),(c,a),(b,d),(d,b),(c,d),(d,c),\\ (e,f),(f,e),(f,g),(g,f),\\ (h,j),(j,h),\\ (k,l),(l,k),(k,m),(m,k),(l,m),(m,l)\}$$

   Here, of course, the key is that the relation be symmetric, as per the definition above.

2. [5 p] Define, for any such graph (V, R) (not just for the example above!), the set D of dead-ends, i.e. the set of all villages one cannot leave (i.e. go to different village) at all by road, or by at most one road. In the example, this set would be $D = \{e, g, h, j, n\}$.

$$D = \{v \in V : \#R(v) \leq 1\}$$

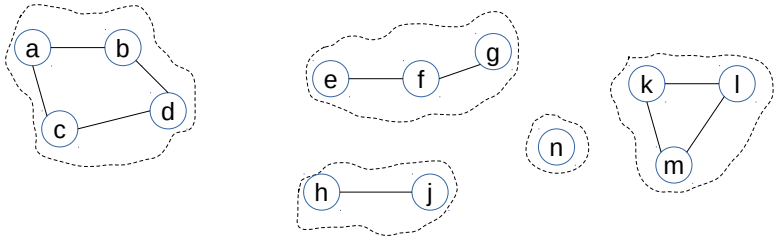


Figure 2: Grouping villages into islands

2 [10 p]

Suppose the villages and roads in (V, R) from task 1 are located on a set of islands, and each island corresponds exactly to the villages that can be reached from one another by road (i.e. all the villages on an island are connected by roads, not necessarily directly), as shown for our example in Figure 2.

Define for any such (V, R) the set A of islands, that is a set of sets of villages that can reach each other by road. In the example, we would have

$$A = \{\{a, b, c, d\}, \{e, f, g\}, \{h, j\}, \{k, l, m\}, \{n\}\}.$$

$$A = \{R[\{v\}] : v \in V\}$$

also, just for example,

$$\{R^+(v) \cup \{v\} : v \in V\}$$

The key here was to compute, for every vertex v, the set of vertices that can be reached from it, by iterating R starting from v (and to not forget to include the vertex itself, which is important if one uses the transitive closure R^+ of R, and the vertex is isolated like the village n in the example). The easiest way to do this is arguably a closure, but there are of course other options. Doing this for every vertex will typically compute the same set multiple times, because (due to symmetry), every vertex within an "island" can reach every other on the same island. However, adding the same set multiple times will not change the result, so there is no problem.

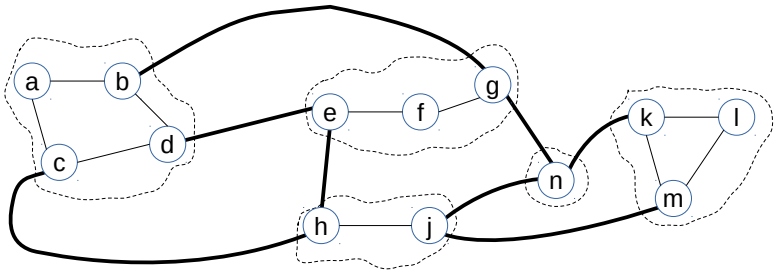


Figure 3: Our example villages and roads on the archilepago, with ferry connections between islands drawn in bolder lines.

3 [10 p]

Suppose that in addition to the villages in V and the symmetric relation of the roads connecting them R from task 1 we also have another *symmetric* relation $F \subseteq V \times V$ of ferry connections connecting villages on different islands, as in Figure 3. We assume that $F \cap R = \emptyset$, that is that there is no ferry connecting villages that are also connected by a road.

Define the set $Ports$ of ports, that is villages that have at least one ferry connection. In the example, this would be $Ports = \{b, c, d, e, g, h, j, k, m, n\}$.

$Ports =$ $F(V)$

also

$\text{dom } F$ or $\text{rng } F$ or $\{v \in V : F(v) \neq \emptyset\}$ and many others

4 [35 p]

Suppose the villages V, the roads $R \subseteq V \times V$, and the ferry connections $F \subseteq V \times V$, as in task 3. The combined connections between all the villages, using roads and ferries, is the relation $E = R \cup F$. (Recall that R and F are disjoint, so $R \cap F = \emptyset$.)

We want to define a function $P : V \times V \longrightarrow \mathcal{P}(V^*)$ such that $P(v_1, v_2)$ returns all cycle-free paths from v_1 to v_2 in E, that is all paths such that every vertex (village) occurs at most once in it. If $v_1 = v_2$, it returns a set containing only the empty path from v_1 to itself, that is $P(v_1, v_1) = \{v_1\}$.

We define P using a helper function $P' : V^* \times V \times V \longrightarrow \mathcal{P}(V^*)$ that builds the path as it searches for its destination:

$$P : V \times V \longrightarrow \mathcal{P}(V^*)$$
$$v_1, v_2 \mapsto P'(\varepsilon, v_1, v_2)$$

1. [25 p] Define P'. You might find it useful to talk about the set of all vertices in a sequence of vertices (that is, in a path) – if $p \in V^*$, then you can use $\text{set}(p)$ to describe the set of all vertices that occur in p.

$$P' : V^* \times V \times V \longrightarrow \mathcal{P}(V^*)$$

$$p, v, w \mapsto \begin{cases} \{pv\} & \text{for } v = w \\ \bigcup_{q \in E(v) \setminus \text{set}(pv)} P'(pv, q, w) & \text{for } v \neq w \end{cases}$$

In this graph algorithm problem, the keys are (a) to figure out when to produce an output path (here: when v and w are equal) and terminate, (b) which direction(s) to continue the recursion into (here: $E(v)$), and (c) which directions to NOT go into (in this case: all the vertices seen up to this point, i.e. $\text{set}(pv)$). Notice that in the above solution, (b) and (c) are combined, by ensuring that for any call $P'(p, v, w)$, it is always true that $v \notin \text{set}(p)$ – which means we do not need to specially test for that case.

As is always the case with problems like this one, there are many ways in which to solve them. A common alternative, which separates dealing with (b) and (c) into distinct cases, went something like this:

$$p, v, w \mapsto \begin{cases} \{pv\} & \text{for } v = w \\ \bigcup_{q \in E(v)} P'(pv, q, w) & \text{for } v \neq w, v \notin \text{set}(p) \\ \emptyset & \text{for } v \neq w, v \in \text{set}(p) \end{cases}$$

Here, it is not always true that $v \notin \text{set}(p)$, so we need to test this explicitly and catch a vertex that has already been visited. This in done in the last case, where we simply return an empty set of paths. In the first solution, P' would not be called on such a vertex, but the result is the same.

2. [10 p] Define the set $\Pi \subseteq V^*$ of all non-cyclic paths in (V, E), as computed by P:

$$\Pi = \bigcup_{v_1, v_2 \in V} P(v_1, v_2)$$

Other options include $P(V \times V)$ and things like $\{p : \exists v_1, v_2 \in V \ (p \in P(v_1, v_2))\}$ etc.

5 [20 p]

Suppose we have a function $d : E \longrightarrow \mathbb{R}^+$ from the combined connections to the positive real numbers, signifying for each $(v_1, v_2) \in E$ the time it takes to travel from v_1 to v_2 (by either road or ferry, depending on whether $(v_1, v_2) \in R$ or $(v_1, v_2) \in F$). For any $(v_1, v_2), (v_2, v_1) \in E$, it is NOT guaranteed that $d(v_1, v_2) = d(v_2, v_1)$.

Given a path $v_0 v_1 v_2 ... v_n$ of length n in E, we want to measure it regarding the total time it takes and also the number of ferry connections in the path. The function $M : \Pi \longrightarrow \mathbb{N} \times \mathbb{R}_0^+$ computes for each non-cyclic path $p \in \Pi$ (the set of all non-cyclic paths from the previous task) a pair (n, r), where n is the number of ferry connections and r the sum of the travel time of all the connections, by road or ferry, in the path.

For every $v \in V$, applying M to the empty path v results in $(0, 0)$, i.e. $\forall v \in V \ (M(v) = (0, 0))$.

Define M.

$M : \Pi \longrightarrow \mathbb{N} \times \mathbb{R}_0^+$

$$p \mapsto \begin{cases} (0,0) & \text{for } p = v \\ (n+1, r + d(v_1, v_2)) & \text{for } p = v_1 v_2 q \wedge (v_1, v_2) \in F \wedge M(v_2 q) = (n, r) \\ (n, r + d(v_1, v_2)) & \text{for } p = v_1 v_2 q \wedge (v_1, v_2) \in R \wedge M(v_2 q) = (n, r) \end{cases}$$

Hint: you might find it useful to distinguish cases where p has the form v (i.e. it is an empty path), from cases where it has the form $v_1 v_2 q$, i.e. a path with at least two vertices (i.e. of length at least 1).

This is a problem involving recursion (or otherwise iterating) over strings/finite sequences, in this case sequences representing paths. The first point here, and one that most answers got right, is the realization that one needs to distinguish between “ferry” and “road” edges connecting vertices in the path. The second is that while we need to look at the first two vertices (so we can figure out what kind of edge connects them), we must only “peel off” the first vertex, and keep recurring on the path beginning with the second. In other words, in the above formulation, the recursive call needs to be $M(v_2 q)$ rather than just $M(q)$.

Some solutions tried to get away with a non-recursive way of computing the two components of the result, which is okay if it is done correctly. Similarly, some solutions used position-wise addition on pairs, which is fine (some of you had asked me about this during the exam), e.g. like so:

$$p \mapsto \begin{cases} (0,0) & \text{for } p = v \\ (1, d(v_1, v_2)) + M(v_2 q) & \text{for } p = v_1 v_2 q \wedge (v_1, v_2) \in F \\ (0, d(v_1, v_2)) + M(v_2 q) & \text{for } p = v_1 v_2 q \wedge (v_1, v_2) \in R \end{cases}$$

There was considerable variation in the way people expressed manipulations of strings, ranging from the solution above to little bits of Clojure code involving `drop` and indexing into strings, like $p[0]$ and $p[1]$. I accepted all of those (if they were otherwise correct), even when they seemed to confuse strings and sets (assuming, again, that the meaning was otherwise clear and correct).

6 [15 p]

With $P : V \times V \longrightarrow \mathcal{P}(V^*)$ from task 4 we can compute the set $P(v_1, v_2)$ of all non-cyclic paths between two villages v_1 and v_2, and with $M : \Pi \longrightarrow \mathbb{N} \times \mathbb{R}_0^+$ from task 5 we can measure each path in terms of the number of ferry connections involved and its total time.

We are interested in the set of "best" paths, which we define as follows:

Suppose $M(p_1) = (n_1, r_1)$ and $M(p_2) = (n_2, r_2)$.
Path p_1 is "better" than p_2. written as $p_1 \prec p_2$, iff $(n_1 < n_2) \vee (n_1 = n_2 \wedge r_1 < r_2)$.

Using the function P defined in task 4, define the function $\hat{P} : V \times V \to \mathcal{P}(V^*)$ such that $\hat{P}(v_1, v_2)$ returns the set of best paths between v_1 and v_2.

Note:

1. $\forall v_1, v_2 \in V \ (\hat{P}(v_1, v_2) \subseteq P(v_1, v_2))$
2. $\forall v_1, v_2 \in V \ \forall p, q \in \hat{P}(v_1, v_2) \ (M(p) = M(q))$

$\hat{P} : V \times V \longrightarrow \mathcal{P}(V^*)$

$v_1, v_2 \mapsto \{p \in P(v_1, v_2) : \neg\exists q \in P(v_1, v_2) \ (q \prec p)\}$

A key to this problem was the realization that different paths might have the same M score, so one cannot make it a condition that $\forall q \in P(v_1, v_2)(p \prec q)$ – for one thing, this will never be true, since $p \in P(v_1, v_2)$ and $p \not\prec p$, but it is not much better to require that $\forall q \in P(v_1, v_2) \setminus \{p\}(p \prec q)$, since there might be other paths with the same M score.

Rules

Things you CAN use during the exam.
Any written or printed material is fine. Textbook, other books, the printed slides, handwritten notes, whatever you like. You may use electronic versions of the material, e.g. PDFs and the like.

In any case, it would be good to have a source for the relevant definitions, and also for notation, just in case you don't remember the precise definition of everything we discussed in the course.

Things you CANNOT use during the exam.
Any communication facility, other than to interact with the examiner.

The exam is ongoing from the moment you receive this document to the moment you submit your answers. **During that time, you must not communicate with anybody other than the examiner and you must not solicit or accept help from any third party with any part of the exam.**

Instructions

- This is a take-home exam. This document was sent to you electronically as a PDF.
- You can print out this document and write your answers in the appropriate spaces. If you do...
  - Fill out the first page and sign.
  - Put your personnummer and name on the subsequent pages in the header.
- If printing is not an option, you can answer the questions on empty sheets of paper. If you do...
  - Make sure to include, on your first page, your name, personnummer, and signature.
  - Mark every subsequent page with your personnummer and name.
  - You need not reproduce the left-most column of the tables in your answers. In that case, your answers will be matched to the questions in the table row by row, so make sure you match the order of your answers to that of the questions.
- Once finished, scan or otherwise photographically capture the pages and produce a PDF from them (using software such as Office Lens, for example).
- The **name of the PDF file** must be your personnummer followed by "`.pdf`", i.e. it has the format

  `yymmdd-nnnn``.pdf`
- **Return the PDF with your answers by replying to the email that you received the question sheet in**. The subject line must include "`[EDAA40 Exam]`" (without the quotes). Do not forget to attach the file.
- If you have questions for the examiner during the exam, contact him by phone first (you will receive the phone number in the email with the exam).

Good luck!

Programming contest

Were you member of a group that qualified for the EDAA40 programming contest this year?

	I was in a group that qualified but did not win.
	I was in the group that qualified and won.
	None of the above.

(please tick the appropriate box)

Group name (if applicable): _____

1 [15 p]

Suppose we have a tree (T, R) with root $a \in T$.

1. [5 p] Define a function $N : \mathbb{N} \longrightarrow \mathcal{P}(T)$ which computes, for every natural number n, the set of all nodes in T that have exactly n children.

   $N : \mathbb{N} \longrightarrow \mathcal{P}(T)$

   $n \mapsto \{t \in T : \#R(t) = n\}$

2. [5 p] Give an expression evaluating to $C_{\max}$, the maximal number of children of any node in the tree.

   $C_{\max} = \max\{\#R(n) : n \in T\}$

3. [5 p] Give an expression computing the node height H of the tree.

   H = $\max\{\# \ R^{-1}[\{n\}] : n \in T\}$

   Many solutions used a helper function to recursively compute the height, which is of course a valid answer (if done correctly).
   The solution above exploits the fact that the node height of a tree is the same as the size of the largest set of ancestors of a node (including the node itself!). One can also get there with the transitive closure of R^{-1}, but in that case special care must be taken to make sure that the result is the node height (by adding 1, for example).

2 [40 p]

Suppose we have a graph (V, E).
You can reuse previous answers. If they are incorrect, it will not affect the answer using them, i.e. I will assume you are using a correct answer to the previous question.

1. [5 p] A vertex $w \in V$ is said to be *reachable* from another vertex $v \in V$ iff there exists a path from v to w. (This includes empty paths, so every vertex is always reachable from itself.)
   Define the relation $R \subseteq V \times V$, such that $(v, w) \in R$ iff w is reachable from v.

   $R = E^+ \cup \{(v, v) : v \in V\}$

   The most common mistake on this task was omitting the second part, i.e. making sure that the reachability relation is reflexive.

2. [5 p] What properties does R have?
   Tick the corresponding box under "always", if R has the property for every graph (i.e., for every possible V and E defining a graph), under "never" if it does NOT have that property for any graph, and under "sometimes" if it has that property for at least one graph and does not have the property for at least one graph.
   Mark the corresponding box under "no answer" if you prefer to not give an answer. See below about how these tables are scored. (1 point per answer)

	always	**sometimes**	**never**	**no answer**
reflexive over V	X			
transitive	X			
symmetric		X		
antisymmetric		X		
asymmetric		X		

   About scoring these tables:

   Every correct answer **adds** the indicated number of points per answer, while every false answer **deducts** the same number of points. Marking "no answer" (or simply not marking any box in a given row) does not change the point score, so it counts as 0. Should the total score for the table be negative, it is counted as zero (0).

3. [5 p] Define the relation $M \subseteq V \times V$, such that $(v, w) \in M$ iff v is reachable from w and w is reachable from v:

   $M = R \cap R^{-1}$

   Many answers look more like $\{(v, w) \in V^2 : vRw \wedge wRv\}$, which is correct and a perfectly good answer.

4. [5 p] What properties does M have?
   Tick the corresponding box under "always", if M has the property for every graph (i.e., for every possible V and E defining a graph), under "never" if it does NOT have that property for any graph, and under "sometimes" if it has that property for at least one graph and does not have the property for at least one graph.
   Mark the corresponding box under "no answer" if you prefer to not give an answer. (1 point per answer)

   The correct answers to this table include M being reflexive, transitive, and symmetric, which means it is an equivalence relation. Realizing this can be helpful in answering the next question.

	always	**sometimes**	**never**	**no answer**
reflexive over V	X			
transitive	X			
symmetric	X			
antisymmetric		X		
asymmetric		X		

5. [10 p] A set of vertices $C \subseteq V$ is called a *strongly-connected component* of this graph iff (a) for any two vertices $v, w \in C$, v is reachable from w and w is reachable from v, and (b) it is maximal, i.e. there is no set C' with $C \subset C' \subseteq V$ such that for any two elements $v, w \in C'$, v is reachable from w and w is reachable from v.
   Define the set $S \subseteq \mathcal{P}(V)$ of **all** strongly-connected components of this graph.

   $S = V/M$

   As can be seen in the previous task, M is an equivalence relation, and the strongly connected components are its equivalence classes, so one way of defining S is as the quotient set of V and M.

   A common way of answering this was $\{M(v) : v \in V\}$, which is also correct.

6. [10 p] Suppose S is the set of strongly-connected components of some graph (V, E).
   Judge the truth of the following formulae.
   Tick the corresponding box under "always", if the formula is true for all graphs, the one under "never" if there is no graph for which this formula is true, and the one under "sometimes" if there is at least one graph for which it is true and at least one for for it is false.
   Mark "no answer" if you prefer not to give an answer. (2 points per answer)

	always	**sometimes**	**never**	**no answer**
$V = \bigcup_{a \in S} a$	X			
$\forall r \in S, s \in S\ (r \cap s = \emptyset)$		X		
$\forall r \in S, s \in S\ (\exists n \in r \cap s\ (n = n))$		X		
$\exists r \in S, s \in S\ (r \cap s \neq \emptyset)$		X		
$\forall s \in S\ (s \neq \emptyset)$	X			

3 [20 p]

Suppose we have an surjection $h : A \twoheadrightarrow C$ which we know to be the result of composing $f : A \longrightarrow B$ and $g : B \longrightarrow C$, i.e. $h = g \circ f$, but we don't know those two functions, nor do we know B. Knowing that h is surjective, what can we say about f and g?

In the following, you will be given a few statements. You are asked to **decide whether they follow from the fact that h is surjective**. If you think a statement does follow from that, prove it. Otherwise, show a counterexample. A counterexample consists of definitions of A, B, C, and f, g, and h, such that the h is surjective, $h = g \circ f$, and the statement is not true. Use proper definitions, not pictures.

1. [5 p] f must be surjective (circle correct answer). YES (NO)
   If yes, prove it, if no, find a counterexample.

   $A = \{a\}$
   $B = \{b_1, b_2\}$
   $C = \{c\}$
   $f = \{(a, b_1)\}$
   $g = \{(b_1, c), (b_2, c)\}$
   $h = \{(a, c)\}$

2. [5 p] g must be surjective (circle correct answer). (YES) NO
   If yes, prove it, if no, find a counterexample.

   To show: $g \circ f$ surjective implies that g is surjective, i.e.
   $h(A) = g \circ f(A) = C \rightarrow g(B) = C$.

   Since h is surjective, it is the case that $h(A) = g \circ f(A) = g(f(A)) = C$.

   Note that for any sets $Y_1, Y_2 \subset X$ and any function $m : X \longrightarrow X'$, it is the case that $Y_1 \subseteq Y_2 \rightarrow m(Y_1) \subseteq m(Y_2)$.

   Thus, since $f(A) \subseteq B$, it follows that $C = g(f(A)) \subseteq g(B)$, and also, since $g : B \longrightarrow C$, we have $g(B) \subseteq C$. Therefore, $g(B) = C$.

3. [5 p] It is always true that $\#A \geq \#B$. YES NO
   If yes, prove it, if no, find a counterexample.

   See counterexample in subtask 1.

4. [5 p] It is always true that $\#B \geq \#C$. YES NO
   If yes, prove it, if no, find a counterexample.

   To show that $\#B \geq \#C$, we must show the existence of a surjection $B \twoheadrightarrow C$. This was done in subtask 2, where $g : B \longrightarrow C$ was shown to be surjective.

4 [25 p]

Suppose a graph (V, E).

You may remember from Lab 5 that a Hamiltonian path was path in the graph in which every vertex was visited exactly once.

By contrast, an *Eulerian path* is a path that uses every *edge* exactly once. A given vertex may occur multiple times in an Eulerian path (since any number of edges can start or end at that vertex), but every edge must be used exactly once.

As with Hamiltonian paths, a given graph can have zero, one, or more Eulerian paths. This task is about **computing the set of all Eulerian paths** of the graph (V, E). We do this in two steps.

1. [15 p] First, we create a function $P : V \longrightarrow \mathcal{P}(V^*)$ that computes for every vertex $v \in V$ the set of all Eulerian paths starting at v.

   It does so using a helper function $P' : V \times V^* \times \mathcal{P}(V \times V) \longrightarrow \mathcal{P}(V^*)$ as follows:

   $$P : V \longrightarrow \mathcal{P}(V^*)$$
   $$v \mapsto P'(v, \varepsilon, E)$$

   Define P'.

   The parameters v, p, and D mean the following:
   v : the current vertex,
   p : the path up to this point, not including v,
   D : the set of edges that haven't been used yet.

   $$P' : V \times V^* \times \mathcal{P}(V \times V) \longrightarrow \mathcal{P}(V^*)$$

$$v, p, D \mapsto \begin{cases} \{pv\} & \text{for } D = \emptyset \\ \bigcup\limits_{w \in D(v)} P'(w, pv, D \setminus (v, w)) & \text{for } D \neq \emptyset \end{cases}$$

2. [5 p] Now define $Q \subseteq V^*$, the set of all Eulerian paths in the graph:

$$Q = \bigcup_{v \in V} P(v) \quad \text{or simply} \quad P(V)$$

3. [5 p] In order to ensure that P' terminates, we require a **well-founded strict order** $\prec$ on its domain $V \times V^* \times \mathcal{P}(V \times V)$, such that for any (v, p, D) that P' is called on, it will only ever call itself on $(v', p', D') \prec (v, p, D)$. Define such an order:

$$(v', p', D') \prec (v, p, D) \iff D' \subset D$$

Remember: A correct answer to this question must have three properties.

1. It must be a strict order on $V \times V^* \times \mathcal{P}(V \times V)$.
2. It must be well-founded, i.e. there cannot be an infinite descending chain in that order.
3. Your definition of P' must conform to it, i.e. any recursive call in it must be called on a smaller (according to the order) triple of arguments.

Rules

Things you CAN use during the exam.
Any written or printed material is fine. Textbook, other books, the printed slides, handwritten notes, whatever you like. You may use electronic versions of the material, e.g. PDFs and the like.

In any case, it would be good to have a source for the relevant definitions, and also for notation, just in case you don't remember the precise definition of everything we discussed in the course.

You can use Clojure to test or validate your definitions, as long as all run is code belonging to the basic Leiningen distribution or that you wrote during the exam and do not access other software tools.

Things you CANNOT use during the exam.
Any communication facility, other than to interact with the examiner.

The exam is ongoing from the moment you receive this document to the moment you submit your answers. **During that time, you must not communicate with anybody other than the examiner and you must not solicit or accept help from any third party with any part of the exam.**

FAQ

Can I make auxiliary/helper definitions?
Yes.

Can I use definitions from previous tasks?
Yes. If you do, include a short note referencing the task you take the definition from.

Should you have made an error in the previous definition, it will **not** affect the score on a task in which you use it. In other words, if you use previous definitions, I will, for the purpose of grading the answer using them, assume that they are correct.

Do I need to provide only the answer or also the calculations I performed to get to it?

Unless I specifically ask for the path to an answer, the answer itself is sufficient.

Instructions

- This is a take-home exam. This document was sent to you electronically as a PDF.
- Print out this document and write your answers in the appropriate spaces.[1] You can use additional sheets if you need to.
    - Fill out the first page and sign.
- Once finished, scan or otherwise photographically capture the pages and produce a PDF from them (using software such as Office Lens, for example). **Include your photo ID on the first page.**
- The **name of the PDF file** must be your personnummer followed by "`.pdf`", i.e. it has the format

  yymmdd`-`*nnnn*`.pdf`
- **Return the PDF with your answers by replying to the email that you received the question sheet in**. The subject line must include "`[EDAA40 Exam]`" (without the quotes). Do not forget to attach the file. Make sure to **include the PDF as an email attachment** – do NOT send a link to your answers.
- If you have questions for the examiner during the exam, contact him by phone first (you will receive the phone number in the email with the exam).

Good luck!

Key points:

- create a legible PDF, name it with your personnummer in the above format
- attach the PDF to the email – no links to hosted files etc.
- return email to the address it came from, with [EDAA40 Exam] in the subject

1 If printing is not an option, you can answer the questions on empty sheets of paper. If you do, make sure to include, on your first page, your name, personnummer, and signature, **and scan it with your photo ID**.

- You need not reproduce complete multiple-choice tables in your answers. In that case, your answers will be matched to the questions in the table row by row, so make sure you match the order of your answers to that of the questions.

Programming contest

Were you member of a group that qualified for the EDAA40 programming contest this year? This is where you claim your bonus points.

	I was in a group that qualified but did not win.
	I was in the group that qualified and won.
	None of the above.

(please tick the appropriate box)

Group name (if applicable): _____

Some conventions you may use in the exam:

For any endorelation $R \subseteq A \times A$ and any set $X \subseteq A$, you can use $R^0(X) = X$ and for any $n \in \mathbb{N}$, $R^{n+1}(X) = R(R^n(X))$. Note that this means that $R^1(X) = R(X)$, i.e. the image of X under R.

In situations when ∞ is a meaningful value, you can use $\min\{\infty\} = \min \emptyset = \infty$, as well as $\min(S \cup \{\infty\}) = \min S$ and $x + \infty = \infty$ for any x.

About scoring multiple choice tables:

Every correct answer **adds** the indicated number of points per answer, while **every incorrect answer deducts the same number of points**. Marking “no answer” (or simply not marking any box in a given row) does not change the point score, so it counts as 0. Should the total score for the table be negative, it is counted as zero (0).

1 [30 p]

Suppose a directed graph (V, E). Assume that $V = \text{dom } E \cup \text{rng } E$. We say that an edge $(v, w) \in V$ should be ... $\in E$ is *going away from* v and *coming into* w.

1. [5 p] Define the set I of vertices $v \in V$ that have at least one edge going away from and no edges coming into them.

   $I = \text{dom } E \setminus \text{rng } E$

2. [5 p] Define the set T of vertices that have at least one edge coming into and no edges going away from them.

   $T = \text{rng } E \setminus \text{dom } E$

3. [5 p] Define the function $d : \mathcal{P}(V) \times \mathcal{P}(V) \longrightarrow \mathbb{N} \cup \{\infty\}$ such that $d(X, Y)$ is the length of the shortest path from any vertex in X to any vertex in Y. If there is no such path, that distance is ∞. If $X \cap Y \neq \emptyset$, it is 0.

   $d : \mathcal{P}(V) \times \mathcal{P}(V) \longrightarrow \mathbb{N} \cup \{\infty\}$

   $X, Y \mapsto \min\{n : E^n(X) \cap Y \neq \emptyset\}$

   Many answers used recursion of some kind. Note that any recursive formulation would have to handle the case that a path does not exist, for example like this

   $$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E(X) \subseteq X \\ 1 + d(X \cup E(X), Y) & \text{otherwise} \end{cases}$$

   Since d's arguments are already sets, there is no need for a helper function, one can simply accumulate the vertices by growing the first argument. Finiteness of V then guarantees termination.

   But accumulate the vertices one must. An alternative is to explicitly test whether the function will terminate, like so:

$$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E^+(X) \cap Y = \emptyset \\ 1 + d(E(X), Y) & \text{otherwise} \end{cases}$$

On the first call, this tests whether repeated applications of E will eventually yield a vertex in Y, and only proceeds with the recursive call if it does. Try to see what a well-founding for the recursion of this function might look like. However, while the above function always terminates, a very similar one does not:

$$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ \infty & \text{if } E^+(X) \cap Y = \emptyset \\ 1 + \min\ \{d(\{x\}, Y) : x \in E(X)\} & \text{otherwise} \end{cases}$$

Keeping previous vertices around to see whether new vertices can be reached or testing explicitly is necessary, however. Consider these two incorrect implementations:

a) $$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ 1 + d(E(X), Y) & \text{otherwise} \end{cases}$$

b) $$X, Y \mapsto \begin{cases} 0 & \text{if } X \cap Y \neq \emptyset \\ 1 + d(E(X) \setminus X, Y) & \text{otherwise} \end{cases}$$

The first looks at the vertices that can be reached from the current set of vertices, the second the front of "newly discovered" vertices $E(X) \setminus X$, until it intersects with Y. Either won't terminate if Y cannot be reached from X. Try running them on $(\{a, b\}, \{(a, a), (b, a)\})$ with $X = \{a\}$ and $Y = \{b\}$.

4. [5 p] Define the $M \subseteq V$ of vertices $v \in V$, such that the shortest path from any vertex in I to v is the same length as the shortest path from v to any vertex in T.

$M = \{v \in V : d(I, \{v\}) = d(\{v\}, T)\}$

5. [5 p] Define a function $e : V \longrightarrow \mathbb{N}$ that maps every vertex $v \in V$ to the number of edges it occurs in, i.e. that are coming into v or going away from v or both.

   $e : V \longrightarrow \mathbb{N}$

   $v \mapsto \#\{(a, b) \in E : a = v \vee b = v\}$

   Many answers used the cardinalities of $E(v)$ and $E^{-1}(v)$ to compute the number of edges. In that case care must be taken to accurately account for a possible self-loop (v, v) not be counted twice. Another approach was to take the cardinality of the union of those sets, sometimes reusing the answer to the next task. However, consider the graph $(\{a, b, c\}, \{(a, b), (b, a), (b, c), (c, b)\})$ and the value for $e(b)$, which should be 4.

   That example also puts the nail in the coffin for this solution:

   $$v \mapsto \#E(v) + \#E^{-1}(v) - \#(E(v) \cap \#E^{-1}(v))$$

   As in this case $E = E^{-1}$, and $E(b) = E^{-1}(b) = \{a, c\} = E(b) \cap E^{-1}(b)$, the result would be $e(b) = 2 + 2 - 2 = 2$, which is clearly incorrect.

   Consequently, since this question is about the number of edges that have a certain property, it is easiest to start with the edges in E, filter them by the property, and count the resulting set, rather than look at sets of vertices.

6. [5 p] Define the function $N : V \longrightarrow \mathcal{P}(V)$ that maps every vertex to the set of the vertices connected to it by an edge (in either direction).

   $N : V \longrightarrow \mathcal{P}(V)$

   $v \mapsto E(v) \cup E^{-1}(v)$

2 [40 p]

Suppose a set of vertices V, and two sets of edges $E_1 \subseteq V \times V$ and $E_2 \subseteq V \times V$ between them, such that (V, E_1) and (V, E_2) are two directed graphs with the same set of vertices. In addition, we have two functions $W_1 : E_1 \longrightarrow \mathbb{N}^+$ and $W_2 : E_2 \longrightarrow \mathbb{N}^+$ that assign the edges in each graph positive natural numbers as *edge weights*.

> *An example of such a situation could be cities connected by different kinds of transport, e.g. buses and trains, where the edge weights might measure e.g. the time it takes to use the corresponding transport.*

The goal is to measure a kind of *distance* $d(v, w)$ between any two vertices $v, w \in V$. It is based on the paths we can use to travel von v to w using the edges in both edge sets as follows.

The idea is that we travel from v to w on a path along the edges of both E_1 and E_2, adding the edge weights along that path as we go along. In addition, whenever we leave a vertex on that path using a edge from a different set than the edge that we used to reach it, we add a constant $s \in \mathbb{N}^+$ to the overall path weight as *switching cost*. So, for example, if we reached the current vertex using an edge from E_1, and we leave it using an edge from E_2, we add, in addition to the edge weights, s to the overall distance. If, on the other hand, we arrive at a vertex on an edge from, for instance, E_2, and we leave it again on an edge from E_2, no switching cost is added, and we just accumulate the corresponding edge weights.

We can switch back and forth between using edges from either edge set any number of times, adding s each time we do so.

> *The switching cost might be the additional delay incurred, for example, by walking from the train station to the bus terminal or vice versa. To simplify things, we assume that the switching cost is always the same, irrespective of the vertex the switching occurs on or whether we switch from E_1 to E_2 or vice versa.*

The distance $d(v, w)$ is the smallest path weight including switching costs for any path of the kind described above between v and w. If there is no way to reach w from v using the edges in E_1 and E_2, their distance is $d(v, w) = \infty$. For any vertex $v \in V$, the distance $d(v, v) = 0$.

Note that it is not necessarily the case that $E_1 \cap E_2 = \emptyset$, i.e. some edges (v_1, v_2) may be in both edge sets, and in that case their weights $W_1(v_1, v_2)$ and $W_2(v_1, v_2)$ need not be the same, so it is important to distinguish whether the edge was taken from E_1 or E_2 to properly keep track of weights and switching costs.

> *If you think of it in terms of buses and trains: even if two cities are connected by both a bus and a train, when you calculate the overall time of an itinerary, you need to distinguish the two in order to figure out (a) how long that part of the journey took (that's the potentially different weights of the same edge) and also (b) to figure out*

whether you need to account for trips between bus terminals and train stations (i.e. potential switching costs).

Also note that you cannot make the assumption that it is always better to avoid switching. Whether switching results in a shorter overall distance will always depend on the edge weights and s.

Define the distance function $d : V \times V \longrightarrow \mathbb{N} \cup \{\infty\}$ in the following two steps.

1. [25 p] First, define a helper function recursively

$$d' : V \times V \times \mathbb{N}^+ \times \{1, 2\} \times \mathcal{P}(V) \longrightarrow \mathbb{N}^+ \cup \{\infty\}$$

with the following meaning of the arguments of $d'(v, w, D, n, S)$:

v	is the current vertex
w	is the vertex we want to reach
D	is the distance accumulated so far, i.e. from the initial starting point up to v
n	is 1 or 2 and denotes the edge set we took the edge from with which we reached the current vertex, 1 for E_1 and 2 for E_2
S	is the set of vertices we have visited so far (including the current vertex v, so it is always the case that $v \in S$)

$v, w, D, n, S \mapsto$

$$\begin{cases} D & \text{for } v = w \\ \min\{d'(v', w, D + W_1(v, v'), 1, S \cup \{v'\}) : v' \in E_1(v) \setminus S\} & \\ \cup\{d'(v', w, D + W_2(v, v') + s, 2, S \cup \{v'\}) : v' \in E_2(v) \setminus S\} & \text{for } v \neq w, n = 1 \\ \min\{d'(v', w, D + W_1(v, v') + s, 1, S \cup \{v'\}) : v' \in E_1(v) \setminus S\} & \\ \cup\{d'(v', w, D + W_2(v, v'), 2, S \cup \{v'\}) : v' \in E_2(v) \setminus S\} & \text{for } v \neq w, n = 2 \end{cases}$$

There are various ways to make this more compact. For example, you can use the fact that 3-n yields 2 for n=1 and 1 for n=2 to shorten this somewhat:

$$v, w, D, n, S \mapsto \begin{cases} D & \text{for } v = w \\ \min\{d'(v', w, D + W_n(v, v'), n, S \cup \{v'\}) : v' \in E_n(v) \setminus S\} & \\ \cup\{d'(v', w, D + W_{3-n}(v, v') + s, 3 - n, S \cup \{v'\}) : v' \in E_{3-n}(v) \setminus S\} & \text{for } v \neq w \end{cases}$$

One rather clever solution managed to shorten the expression in the final case to

$$\min\{d'(v', w, D + W_k(v, v') + s \cdot \#(\{k\} \setminus \{n\}), k, S \cup \{v'\}) : k \in \{1, 2\}, v' \in E_k(v) \setminus S\}$$

Key points of a correct solution are the following:

- termination when $v = w$
- recursive calls on d' with (a) the proper weight added to D depending on which edge set the edge was taken from and (b) additionally s added to the weight when that edge set was different from the one v was reached on,
- exploring all possible paths to w (filtered only by the growing S to avoid cycles), returning the minimum weight from all of them.

This last point bears emphasizing. One common mistake was to choose a particular edge from E_1 and E_2 based on their weight and switching cost, leading to many more cases with complicated conditions. However, this choice cannot be made locally, because it affects the subsequent weights one encounters after picking the “cheapest” next step.

Many solutions also explicitly tested for non-reachability of the target vertex and returned ∞ in that case, for example by using a case with a condition like $(E_1 \cup E_2)(v) \setminus S = \emptyset$. While not incorrect (and so no points were deducted from the score), it is redundant given the convention that $\min \emptyset = \infty$. If the rest was done correctly, the sets of recursive calls to d' would eventually become empty, and so $\min$ would automatically produce the desired result.

2. [10 p] Using the helper function d' from the previous step, define the distance function

   $$d : V \times V \longrightarrow \mathbb{N} \cup \{\infty\}$$

   as described above.

$$v, w \mapsto \begin{cases} 0 & \text{for } v = w \\ \min\{d'(v', w, W_1(v, v'), 1, \{v, v'\}) : (v, v') \in E_1)\} & \\ \cup\{d'(v', w, W_2(v_1, v'), 2, \{v, v'\}) : (v, v') \in E_2)\} & \text{otherwise} \end{cases}$$

Some solutions looked like this:

$$v, w \mapsto \min\{d'(v, w, 0, 1, \{v\}), d'(v, w, 0, 2, \{v\})\}$$

That computes a correct result, but technically violates the definition of d', which requires a positive value as a third argument. (A few solutions noticed this and worked around it by passing a 1 into d' and subtracting it again at the end.) I concluded that this hint was too subtle and let it slide, accepting solution that passed 0 to d'.

3. [5 p] In order to ensure that d' terminates, we require a **well-founded strict order** $\prec$ on its domain $V \times V \times \mathbb{N}^+ \times \{1, 2\} \times \mathcal{P}(V)$, such that for any (v, w, D, n, S) that d' is called on, it will only ever call itself on $(v', w', D', n', S') \prec (v, w, D, n, S)$. Define such an order:

$$(v', w', D', n', S') \prec (v, w, D, n, S) \iff S' \supset S$$

Note: You are NOT required to *prove* that the order you define is well-founded, it suffices that it is. More specifically, a correct answer to this question must have three properties.

1. It must define a strict order on $V \times V \times \mathbb{N}^+ \times \{1, 2\} \times \mathcal{P}(V)$.
2. It must be well-founded, i.e. there cannot be an infinite descending chain in that order.
3. Your definition of d' must conform to it, i.e. any recursive call in your definition must be called on a smaller (according to the order) quintuple of arguments.

3 [30 p]

Given is a directed graph (V, E), a function $\lambda : E \longrightarrow A$, assigning each edge a label from a set A, and another function $W : E \longrightarrow \mathbb{R}^+$, assigning each edge a non-negative real number (its *weight*).

For any set of labels $S \subseteq A$, let $E_S = \lambda^{-1}(S) = \{e \in E : \lambda(e) \in S\}$, i.e. the set of edges whose label is in S.

With this, we define a family of distance functions $d_X : V \times V \longrightarrow \mathbb{R}^+ \cup \{\infty\}$ for every $X \subseteq A$, as follows:

$$d_X : V \times V \longrightarrow \mathbb{R}^+ \cup \{\infty\}$$

$$v, w \mapsto d'_X(v, w, \emptyset)$$

using a family of helper functions d'_X defined as follows:

$$d'_X : V \times V \times \mathcal{P}(V) \longrightarrow \mathbb{R}^+ \cup \{\infty\}$$

$$v, w, S \mapsto \begin{cases} 0 & \text{for } v = w \\ \min(\{d'_X(v', w, S \cup \{v\}) : v' \in E_X(v) \setminus S\} & \\ \quad \cup \{W(v, v') + d'_X(v'w, S \cup \{v\}) : v' \in E_{A \setminus X}(v) \setminus S\}) & \text{otherwise} \end{cases}$$

We now use those distance functions to define a relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ on the subsets of A:

$$R = \{(X, Y) \in \mathcal{P}(A) \times \mathcal{P}(A) : \forall v \in V, v' \in V \ (d_X(v, v') \leq d_Y(v, v'))\}$$

In order to fill in the tables below, it helps to get a deeper understanding of these definitions. $d_X(v, w)$ is the smallest path distance between v and w, where the weights of the edges labeled with a label in X, i.e. the edges in E_X, are NOT counted, let's say those edges are *null-weighted*. This means that larger label sets can never make the path weight larger, only smaller, because adding labels to the label set means that all null-weighted edges remain null-weighted, plus whatever edges are labeled with the labels that were added. So $X \subseteq X' \rightarrow d_{X'}(v, w) \leq d_X(v, w)$ for any two $v, w \in V$.

Therefore, $(X, Y) \in R$ if and only if null-weighting the edges in E_X always leads to path weights that are not larger than null-weighting the edges in E_Y. Since supersets never make path weights larger, it follows that $X \subseteq X' \rightarrow (X', X) \in R$. Note that this means that for any $X \in \mathcal{P}(A)$, it is always true that $(A, X) \in R$ and $(X, \emptyset) \in R$. Also, for any $X \in \mathcal{P}(A)$, $(X, X) \in R$.

A few corner cases that might be interesting to think about:

- C_0 : the empty graph $(\emptyset, \emptyset)$ with an empty label set $A = \emptyset$. (Note that only a graph with no edges permits an empty label set – since $\lambda : E \longrightarrow A$ is a function, $A \neq \emptyset$ as soon as $E \neq \emptyset$.) Then, $\mathcal{P}(A) = \{\emptyset\}$ and $R = \{(\emptyset, \emptyset)\}$.
- C_1: a non-empty graph (V, E) with $V \neq \emptyset \neq E$, with a singleton label set $A = \{L\}$ (which therefore implies that all edges are labeled the same). Then, $\mathcal{P}(A) = \{\emptyset, A\}$ and $R = \{(\emptyset, \emptyset), (A, \emptyset), (A, A)\}$ (convince yourself that that is the case).
- C_2: a non-empty graph (V, E) with $V \neq \emptyset \neq E$, with label set $A = \{L_1, L_2\}$ and a constant labeling function $\lambda : e \mapsto L_1$. In other words, this is like the case C_1, except that we have an "unused" extra label L_2 in the label set. In this case, $\mathcal{P}(A) = \{\emptyset, \{L_1\}, \{L_2\}, A\}$ and

  $$\begin{aligned} R = \{&(A, A), (A, \{L_1\}), (A, \{L_2\}), (A, \emptyset), \\ &(\{L_1\}, A), (\{L_1\}, \{L_1\}), (\{L_1\}, \{L_2\}), (\{L_1\}, \emptyset), \\ &(\{L_2\}, \{L_2\}), (\{L_2\}, \emptyset), \\ &(\emptyset, \{L_2\}), (\emptyset, \emptyset)\} \end{aligned}$$

- C_3: a non-empty, unconnected graph $(V, \emptyset)$ with $V \neq \emptyset$ and some non-empty label set $A \neq \emptyset$. Note that this means that distances are unaffected by the set of labels : $d_X(v, v')$ is 0 if $v = v'$ and ∞ otherwise, for any $X \in \mathcal{P}(A)$. Consequently, $R = \mathcal{P}(A) \times \mathcal{P}(A)$.

1. [10 p] The relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$ depends on the combination of the graph (V, E), the label set A and the labeling and weight functions λ and w.

   In the following table, several properties are given. Tick the corresponding box under "always", if R has the property for every valid combination of graph, label set, and labeling and weight function, under "never" if it does NOT have that property for all such combinations, and under "sometimes" if it has that property for at least one combination and does not have the property for at least another one.
   Mark the corresponding box under "no answer" if you prefer to not give an answer.

   See p. 4 on how these tables are scored.

   [1 point per answer]

		always	**sometimes**	**never**	**no answer**
1	reflexive over $\mathcal{P}(A)$	X			
2	transitive	X			
3	symmetric		X		
4	antisymmetric		X		
5	asymmetric			X	
6	is a strict order			X	
7	is a non-strict order		X		
8	is a total order		X		
9	has a minimum element		X		
10	has a maximum element		X		

   Note: Be sure of the distinction between minimum and minimal, and also between maximum and maximal.

Notes:

1. The condition for $(X, X) \in R$ is that for all $v, v' \in V$ it must be the case that $d_X(v, v') \leq d_X(v, v')$ (vacuously so if $V = \emptyset$), which is clearly the case no matter what X we choose, so R is always reflexive.
2. If $(X, Y) \in R$ and $(Y, Z) \in R$, it means that for any $v, v' \in V$, it is the case that $d_X(v, v') \leq d_Y(v, v')$, and also that $d_Y(v, v') \leq d_Z(v, v')$, so therefore $d_X(v, v') \leq d_Z(v, v')$, and consequently $(X, Z) \in R$. Hence, R is always transitive.
3. In general, R is not symmetric (larger sets tend to lead to smaller distances), but if the label set is empty as in case C_0, it is. Similarly, it is for unconnected graphs, such as case C_3.
4. It would seem that R might be antisymmetric – after all, it is generally not symmetric, but reflexive, and it certainly is antisymmetric in case C_1. But note how in case C_2, $(\{L_1\}, A) \in R$ and $(A, \{L_1\}) \in R$, so it is not always antisymmetric. Neither is it in case C_3.
5. Since R is always reflexive, and never empty, it cannot be asymmetric.
6. For the same reasons, it cannot be a strict order.
7. It is a non-strict order when it is antisymmetric, i.e. sometimes. For example, not in case C_2.
8. It is in cases C_0 and C_1, but in C_2 it is not an order at all.
9. It has a minimum (namely A) when R is an order.
10. It has a maximum (namely $\emptyset$) when R is an order.

2. [20 p] Similar to the previous task, the following table contains statements about the definitions above, such as the distance functions d_X and the relation $R \subseteq \mathcal{P}(A) \times \mathcal{P}(A)$. Whether they are true or false might depend on the specific combination of the graph (V, E), the label set A and the labeling and weight functions λ and w.

   Tick the corresponding box under "always", if the statement is true for every valid combination of graph, label set, and labeling and weight function, under "never" if it is false for all such combinations, and under "sometimes" if it is true for at least one combination and false for at least another one.
   Mark the corresponding box under "no answer" if you prefer to not give an answer.

   See p. 4 on how these tables are scored.

   [2 points per answer]

		always	sometimes	never	no answer
1	$\forall X, Y \in \mathcal{P}(A)\ \forall v, w \in V$ $(d_X(v,w) = \infty \leftrightarrow d_Y(v,w) = \infty)$	X			
2	$\forall X, Y \in \mathcal{P}(A)\ \forall v, w \in V$ $(d_X(v,w) = 0 \leftrightarrow d_Y(v,w) = 0)$		X		
3	$\forall X, Y \in \mathcal{P}(A)\ ((X,Y) \in R \rightarrow X \subseteq Y)$		X		
4	$\forall X, Y \in \mathcal{P}(A)\ (X \subset Y \rightarrow (X,Y) \in R)$		X		
5	$\forall X, Y \in \mathcal{P}(A)\ ((X \cup Y), X) \in R)$	X			
6	$\forall X, Y \in \mathcal{P}(A)\ (X, (X \cup Y)) \in R)$		X		
7	$\forall X, Y \in \mathcal{P}(A)\ ((X \cap Y), X) \in R)$		X		
8	$\forall X, Y \in \mathcal{P}(A)\ (X, (X \cap Y)) \in R)$	X			
9	$\exists X \in \mathcal{P}(A)\ \forall v, w \in V\ (d_X(v,w) = \infty)$		X		
10	$\exists X \in \mathcal{P}(A)\ \forall v, w \in V\ (d_X(v,w) = 0)$		X		

   Hint: In deciding on these properties and statements, it can help to consider "corner cases": very small graphs, very sparse or very connected graphs (i.e. very few or very many edges), graphs where all edge weights are 0, where all edges are labeled the same or all are labeled differently etc.

Notes:

1. The distance functions will only return the distance ∞ if there is no path from the first to the second vertex, so this is always the case irrespective of label set or vertices chosen.
2. This, however, is generally not true: the distance $d_X(v, v')$ can only be zero if $v = v'$ or all edges between the two vertices are labeled by labels in X. However, since this universally quantifies over the vertices, it is true for an empty graph, or for an unconnected graph, in which all distances are 0 or ∞ irrespective of the label set.
3. Generally, this would be false: shorter distances would signify larger label sets. Yet, in case C_0, the conclusion $X \subseteq Y$ is true for all $X, Y \in \{\emptyset\}$, therefore the implication holds, and thus the formula is true.
4. Once again, this formula would tend to be false in general. However, in C_0, the premise $X \subset Y$ is false for all $X, Y \in \{\emptyset\}$, so the implication holds, and therefore the formula is true.
5. As we observed, larger sets will always lead to distances that are not larger, so this is always true.
6. This would conversely tend to be false, but it is true, e.g. in case C_0 of an empty graph, and also case C_3, when the graph is unconnected.
7. This is the dual of 6: it would usually be false, but true for cases like C_0 and C_3.
8. This, in turn, is the dual of 5: smaller sets lead to not-smaller distances, so this is always true.
9. As observed before, the choice of the set X has no bearing on whether the distance $d_X(v, v')$ becomes infinite: if it is infinite for some $X \in \mathcal{P}(A)$, then it is for all of them. This can only be the case if there are no vertices and the inner universal quantifier becomes vacuously true (case C_0). Even in an unconnected non-empty graph (case C_3) the distances of the vertices to themlselves are 0.
10. This is true for graphs in which every vertex can be reached from every other vertex. Then, (at least) the set of all labels would lead to all-zero distances. However, in graphs where some vertices cannot be reached by some other vertices, their distance remains ∞, regardless of the choice of X.