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. $M = f(X)$.

<u>Example:</u> 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):

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

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 $\{v_0, v_1, ..., v_n\}$.

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

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

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] $((p \to q) \,\overline{\wedge}\, (q \leftrightarrow r)) \wedge ((r \to s) \,\overline{\wedge}\, (s \leftrightarrow p))$

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

```
(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
```