1. Analog zur Konstruktion der universellen Turingmaschine könnte man ja auch hoffen einen universellen endlichen Automaten zu konstruieren; d.h. einen endlichen Automaten A_U, der bei Eingabe einer Kodierung w_A eines Automaten A das Wort $w_A\#w$ genau dann akzeptiert, wenn A dieses akzeptiert.
   Zur Einfachheit nehmen wir an, daß wir nur Automaten simulieren mit $\Sigma = \{0, 1\}$, deren Zustände $q_0, q_1, \ldots$ heißen, und die q_0 als Startzustand und q_1 als einzigen Endzustand haben.

   Der universelle Automat darf allerdings ein größeres Alphabet verwenden.

   (a) Erklären Sie, wie Sie einen endlichen Automaten A als Wort w_A kodieren würden.

   (b) Zeigen Sie, daß es keinen universellen endlichen Automaten geben kann.

   Lösung:

   (a) Für einen Automaten A mit Übergangsfunktion δ kodieren wir zunächst jeden Übergang $\delta(q_i, x) = q_j$ durch

   $$\mathrm{bin}(i)\#x\#\mathrm{bin}(j)$$

   und den gesamten Automaten indem wir diese Übergänge mit \$ zu einem Wort w_A "zusammenkleben".

   (b) Nehmen wir an ein solcher Automat existiert, d.h. die Sprache

   $$L = \{\, w_A\$u \mid A \text{ akzeptiert } u \,\}$$

   is regulär. Nach dem Pumpinglemma gibt es ein n mit den üblichen Eigenschaften. Betrachten wir nun den Automaten

   $$A = (\{0, 1\}, \{q_0, \ldots, q_n\}, q_0, \{q_1\}, \delta)$$

   mit $\delta(q_0, 1) = q_2$, $\delta(q_i, 1) = q_{i+1}$ für $2 \leqslant i \leqslant n$ und $\delta(q_n, 1) = q_1$. Dieser Automat akzeptiert genau das Wort 1^n, also ist

   $$w_A\$1^n \in L\ .$$

   Da schon $|w_A| \geqslant n$ ist $|w_A\$1^n| \geqslant n$. Also gibt es wie üblich eine Zerlegung vxu wieder mit den Pumpinglemma Eigenschaften. Nach Voraussetzung ist $x \neq \varepsilon$ und $vw \in L$. Wie man leicht einsehen kann muss $vu = w_{A'}\$1^n$ sein, wobei A' ein Automat ist, der weniger Übergänge als A hat. Daraus folgt leicht, daß A' das Wort 1^n nicht mehr akzeptieren kann (da kein Wort mehr akzeptiert wird). Dies heißt $vu \notin L$; ein Widerspruch.

2. Finden Sie ein `LOOP`-Programm, das mit der Speicherbelegung [] beginnt (d.h. alle Variablen sind mit 0 belegt), und mit einer Speicherbelegung σ endet, so daß $\sigma(x_1) > 1000$. Die Instruktion $x_i \coloneqq x_j + c$ darf allerdings nur in der Form $x_i \coloneqq x_j + 1$ verwendet werden (sonst wäre man ja in einer Zeile fertig). Was ist das kürzeste mögliche solche Programm, das Sie finden können?

Lösung: Viele geschachtelte LOOP Schleifen.

3. Zeigen Sie, daß die folgenden Funktionen LOOP-berechenbar sind:
   (a) Die Differenz $\lambda x.\lambda y.(|x - y|)$
   (b) Die Multiplikation
   (c) Die Division ohne Rest
   (d) Die Mod-Funktion (also der Rest einer Division)

   Lösung:

   (a) Die Idee ist, daß wir uns zuerst die zwei Eingaben merken und als erstes $x_1 - x_2$ (abgeschnittene Subtraktion) und dann $x_2 - x_1$ berechnen (auf den gespeicherten Variablen) und als letztes beide Ergebnisse addieren.

   $x_3 := x_1;$
   $x_4 := x_2;$
   LOOP x_2 DO $x_1 := x_1 - 1$ END;
   LOOP x_4 DO $x_3 := x_3 - 1$ END;
   LOOP x_4 DO $x_1 := x_1 + 1$ END;

   (b)

   $x_3 := x_1;$
   $x_1 := 0;$
   LOOP x_3 DO LOOP x_2 DO $x_1 := x_1 + 1$ END END;

   (c)

   $x_3 := 0;$
   $x_4 := x_1;$
   LOOP x_1 DO
       LOOP x_2 DO $x_4 := x_4 - 1$ END
       IF $x_4 \neq 0$ THEN
           $x_1 := x_4;$
           $x_3 := x_3 + 1;$
   END

   wobei wir IF $x_1 \neq 0$ THEN P als IF $x_1 \neq 0$ THEN $x_1 := x_1$ ELSE P wie in der nächsten Aufgabe beschrieben abkürzen können.
   Dieses Programm berechnet sowohl die Division (Ergebnis in x_3) und den Rest (Ergebnis in x_1)

4. In der Vorlesung haben wir gesehen, wie wir IF $x_i = 0$ THEN P Konstrukte als syntaktischen Zucker zur Sprache LOOP hinzufügen können. Erweitern Sie diesen syntaktischen Zucker zu IF $x_i = 0$ THEN P ELSE Q Statements. Geben Sie ausserdem an, wie man mit IF $x_i = x_j$ THEN P ELSE Q Statements umgehen könnte.

Lösung: Zur Erinnerung: IF $x_i = 0$ THEN P ist wie folgt definiert.

$$\begin{aligned}
&x_j := 1;\\
&\texttt{LOOP } x_i \texttt{ DO } x_j := 0 \texttt{ END};\\
&\texttt{LOOP } x_j \texttt{ DO } P \texttt{ END};\\
&x_j := 0;
\end{aligned}$$

Die Idee können wir nun erweitern zu IF $x_i = 0$ THEN P ELSE Q:

$$\begin{aligned}
&x_j := 1;\\
&x_k := 1;\\
&\texttt{LOOP } x_i \texttt{ DO } x_j := 0 \texttt{ END};\\
&\texttt{LOOP } x_j \texttt{ DO } P; x_k := 0 \texttt{ END};\\
&\texttt{LOOP } x_k \texttt{ DO } Q \texttt{ END};\\
&x_j := 0;\\
&x_k := 0;
\end{aligned}$$

wobei natürlich x_j und x_k frische Variablen sind.
Sei jetzt DIFF das LOOP-Programm von oben, das die Differenz $\lambda x.\lambda y.|x-y|$ berechnet. Dann ist

$$x_k := \mathrm{DIFF}(x_i, x_j); \texttt{IF } x_k = 0 \texttt{ THEN } P \texttt{ ELSE } Q$$

die gesuchte Konstruktion, wobei natürlich x_k wieder eine frische Variable ist.