Quickies

Q.1: nein, es gibt auch unendliche.
Q.2: nein.
Q.3: nein.
Q.4: ja. Man ersetze F durch Q \ F.
Q.5: ja, sogar regulär.
Q.6: nein.

Aufgabe 1

1.1:
- Übergänge können doppelt oder auch gar nicht belegt sein, z.B. S → aA | aB.
- Die Route muss nicht eindeutig sein.

1.2:
Ergänze bei Regeln der Form Nichtterminalsymbol (NTS) geht auf Terminalsymbol (TS) $A \to b$ das implizite Nichtterminalsymbol, das dann auf Epsilon geht.

Prüfe bei jedem NTS auf der linken Seite, ob auf dessen rechter Seite ein TS (mindestens) doppelt vorkommt, z.B. $A \to bA \mid bS$. Wenn nein, dann bist du fertig. Ansonsten:

Wähle einen der Knoten A aus, bei dem doppelte TS vorkommen und fixiere nun auch das TS b.

Sammle zu diesem Paar $A \to b\square$ alle NTS und packe sie in die Menge M.

Also $M = \{X \in V \mid \text{es gibt Regel } A \to bX\}$, im Beispiel: $M = \{A, S\}$.

Wir löschen nun alle Regeln, die bei der Erzeugung von M beteiligt waren und ersetzen sie durch eine neue Regel $A \to bN$ mit einem neuen Knoten N

Jetzt vereinigen wir alle rechten Seiten zu allen NTS aus M und darauf schicken wir einen neuen Knoten N. Im Beispiel also die rechte Seite von A (was jetzt b N ist) und von S.

Wir beginnen von vorne.

1.3:
$<S> \to a<A> \mid b<Y>$ //keine Probleme
$<A> \to a<Y> \mid b<AS>$ //bA und bS werden zu bAS zusammengefasst.
$<AS> \to a<AY> \mid b<ASY>$ //Spaltenweise vereinigen
$<AY> \to \varepsilon \mid a<Y> \mid b<AS>$ // A hat kein Epsilon, daher ist A mit Epsilon neuer Zustand.
$<ASY> \to \varepsilon \mid a<AY> \mid b<ASY>$ //analog, AS hat auch kein Epsilon.
$<Y> \to \varepsilon$ //Die akzeptierenden Endzustände enthalten das Y.
//Zustand <SY> sowie Trap kommen nicht vor.

1.4: php-Implementierung unter https://bit.ly/2G4VFEB

```
programm DEA_konvertierbare_Grammatikanpassung
struct Übergang {             //für L → Rt , „oder“ wird als mehrere Übergänge interpretiert.
    NTS L;
    NTS | null R;             //Regeln der Form L → Rt werden behandelt als L → ε t
    TS | null t;
}
struct Grammatik {
    Menge an NTS V;
    Menge an TS Σ;
    Menge an Übergang T;
    Start-NTS S
}

funktion Grammatikanpassung(Grammatik g) returns Grammatik {
    g = g.clone();                                  //Kopie von g erzeugen

    falls (es keinen Übergang der Form Y → ε und sonst auf garnichts gibt), dann {
        g.V.add(neuer NTS Y);                       //Y wird der akzeptierende Endzustand
        g.T.add(neuer Übergang Y → ε )
    } ansonsten nenne diesen Knoten Y.

    ∀NTS A ∈ g.V {                                  //Schleife über NTS
        g = pruefeUebergangeMitLinkerSeite(g, A);
    }
    return g;
}

funktion pruefeUebergangeMitLinkerSeite(Grammatik g, NTS A) returns Grammatik {
    ∀TS b ∈ g.Σ {
        falls ( A → b in g.T enthalten), dann {
            g.T.entferne ( A → b );
            g.T.add ( A → bY );                     //Y war der oben definierte Endzustand
        }
        Menge P := {Übergang U ∈ g.T | U.L=A ∧ U.t=b};   //d.h. U = A → t❑
        falls ( |P| ≥ 2 ), dann {                   // Problem: Z_0 → zZ_1 | zZ_2
            Menge M := {U.R ∈ V | U ∈ P}  = {X ∈ V | es gibt in T Regel A → b X};
            g.T.entferneAlle (P);                   // Probleme löschen. Menge M behält Infos.
            neuer NTS N;                            // Ersatzknoten N
            g.T.add(neuer Übergang( A → bN ));      // Anbindung an das A
            ∀NTS B ∈ M  ∀Übergang W ∈ g.T mit W.L=B {
                g.T.add(neuer Übergang( N → W.t W.R ));
            }
            falls es in g noch keinen solchen Knoten N gibt, dann {
                g.V.add(N);
                g = pruefeUebergangeMitLinkerSeite(g, N);
            }
        }                                           // endif ( |P| ≥ 2 )
    }                                               // next b
    return g;
}

end programm
```

Aufgabe 2

$$L_1 = \{ww^R \mid w \in \{0,1,2,3,4\}^+ \wedge |w| \leq 4\}$$
$$L_2 = \{a^n b^m b^m a^n \mid n, m \in \mathbb{N}\}$$

Lösung: L_1 ist <u>regulär</u>. L_2 ist <u>nicht regulär</u>.

L_1 ist endlich mit $|L_1| = 780 = \sum_{i=1}^{4} 5^i = \sum_{i=1}^{4}$ Anzahl der Wörter aus 5 Zeichen mit Länge genau i

Jede endliche Sprache ist regulär.

zu L_2: Verwenden des Pumping-Lemmas:

- Annahme L_2 regulär mit Pumping-Konstante p.
- Wähle Wort $\boldsymbol{w} := \boldsymbol{a^p b^p b^p a^p} \in L_2 \wedge |w| = 4p > p$.
- Pumping-Garantien: $// \; links: \Sigma^* \times \mathbb{N}_0 \rightarrow \Sigma^*$, Abbildungsvorschrift klar
  $$\exists xyz = w \wedge 1 \leq |y| \leq |xy| \leq p \wedge xz \in L_2$$
- Weil $|xy| \leq p$ und $\text{links}(w, p) = a^p$ ist $\boldsymbol{xy} \in \boldsymbol{a^*}$. Weil $|y| \geq 1$: $\exists_{1 \leq k \leq p}$: $\boldsymbol{y} = \boldsymbol{a^k}$.
- $L_2 \ni xz = a^{p-k} b^p b^p a^p \notin L_2$ WIDERSPRUCH. Annahme falsch. L_2 nicht regulär.

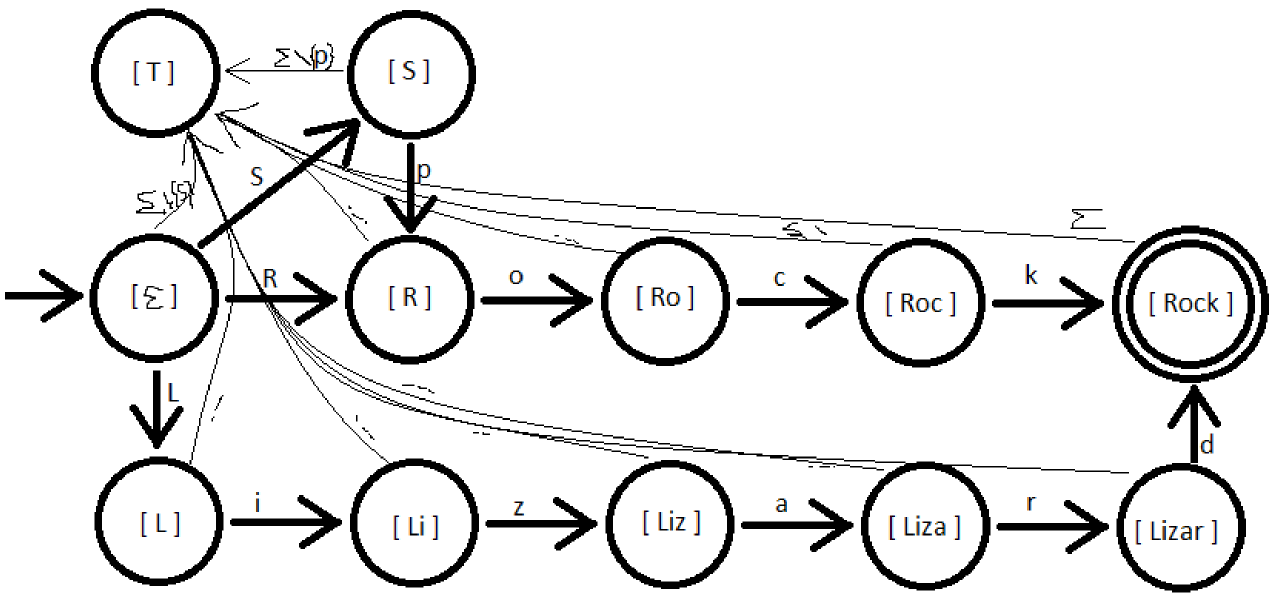
Aufgabe 3

$$L_1 := \{\text{Rock}, \text{Lizard}, \text{Spock}\}$$
$$L_2 := \{w \in \{0,1\}^* \mid 5 \mid \#_1(w)\}$$
$$L_3 := \{w \in \{0,1\}^* \mid \#_1(w) = 5\}$$

Reprä.	**z | xz ∈ L₁**	**z | xz ∉ L₁**
ε	Rock, Lizard, Spock	ε, R, **ock**, L, S, T
R	**ock**	ε, ck, o, S
Ro	ck	ε, k, c, S
Roc	k	ε, S
Rock	ε	S, L, T
S	pock	ε, **ock**

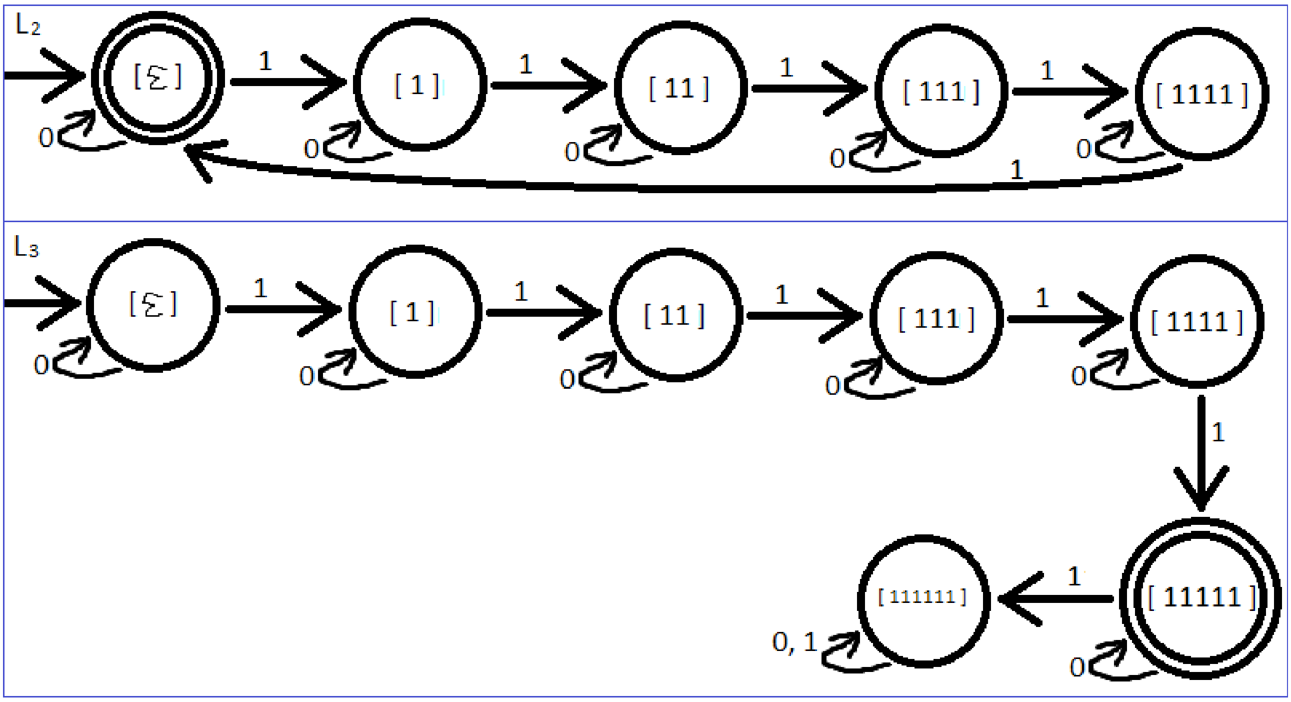
Reprä.	**z | xz ∈ L₁**	**z | xz ∉ L₁**
L	izard	ε, zard, S
Li	zard	ε, ard, S
Liz	ard	ε, rd, S
Liza	rd	ε, d, S
Lizar	d	ε
T //Trap		alles: Σ^*

L_1



Reprä.	$z \mid xz \in L_2$	$z \mid xz \notin L_2$
ε	ε, 11111, 1^{10}	1111, 1
1	1111, 1^9	ε, 111, 1, 0
11	111, 1^8	ε, 11, 1, 0
111	11, 1^7	ε, 1, 0
1111	1, 1^6	ε, 1111, 0
// no trap //		

Reprä.	$z \mid xz \in L_3$	$z \mid xz \notin L_3$
ε	11111, 01001111	ε, 1111, 1, 0
1	1111, 101011	ε, 111, 1, 0
11	111, 1011	ε, 11, 1, 0
111	11, 01010	ε, 1, 0
1111	1, 00001	ε, 1111, 0
11111	ε, 0, 00	1, 11, ...
111111 //trap		alles: Σ^*



Aufgabe 4

$L_1 := \{ww^R \mid w \in \Sigma^*\}$ ist nicht regulär.

Widerspruchsbeweis mit Pumping-Lemma:

- Annahme L_1 regulär mit Pumping-Konstante p. Seien $a, b \in \Sigma, a \neq b$ // $|\Sigma| \geq 2$
- Wähle Wort $\boldsymbol{w := a^p b b a^p} \in L_1 \wedge |w| = 2p+2 > p$.
- Pumping-Garantien: // $links: \Sigma^* \times \mathbb{N}_0 \to \Sigma^*$, Abbildungsvorschrift klar
  $\exists xyz = w \wedge 1 \leq |y| \leq |xy| \leq p \wedge xz \in L_1$
- Weil $|xy| \leq p$ und $\text{links}(w, p) = a^p$ ist $\boldsymbol{xy \in a^*}$. Weil $|y| \geq 1$: $\exists_{1 \leq k \leq p}$: $\boldsymbol{y = a^k}$.
- $L_2 \ni xz = a^{p-k} b b a^p \notin L_2$ WIDERSPRUCH. Annahme falsch. L_2 nicht regulär.

$L_2 := \{0^m 1^n 0^n \mid n, m \in \mathbb{N}\}$ // wobei $0 \notin \mathbb{N}$ ist nicht regulär.

Widerspruchsbeweis mit Pumping-Lemma:

- Annahme L_2 regulär mit Pumping-Konstante p.
- Wähle Wort $\boldsymbol{w := 0 1^p 0^p} \in L_1 \wedge |w| = 2p+1 > p$.
- Pumping-Garantien:
  $\exists xyz = w \wedge 1 \leq |y| \leq |xy| \leq p \wedge xz \in L_2$
- Weil $1 \leq |y| \leq |xy| \leq p$ und $\text{links}(w, p) = 0 1^{p-1}$ ist $\boldsymbol{xy \in 01^*}$.
  Fall 1: $x = \varepsilon$
  $y \in 01^*$, nämlich $\exists_{0 \leq k \leq p-1}: y = 01^k$.
  Dann $L_2 \ni xz = z = 1^{p-k} 0^p \notin L_2$ WIDERSPRUCH
  Fall 2: x beginnt mit 0
  $y \in 1^*$, nämlich $\exists_{1 \leq k \leq p-1}: y = 1^k$ // $k = |y| \geq 1$
  Dann $L_2 \ni xz = 01^{p-k} 0^p \notin L_2$ WIDERSPRUCH
- Alle Fälle führen zum Widerspruch. Annahme falsch. L_2 nicht regulär.