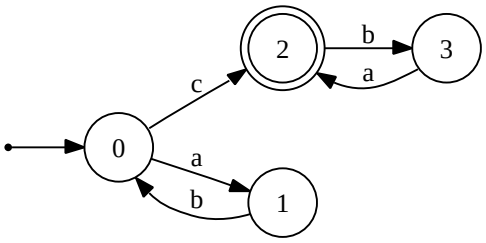
Aufgabe 1.
DFAs zu gegebenen Regulären Ausdrücken. Jeder Automat enthält einen (nicht eingezeichneten) Fehlerzustand, der immer dann angesteuert wird, wenn ein Buchstabe nicht verarbeitet werden kann.

a) $DFA = \{\{0,1,2,3\}, \{a,b,c\}, \delta, 0, \{2\}\}$

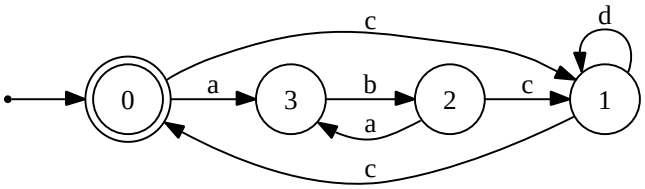
0	x			
1	x	x		
2	x	x	x	
3	x	x	x	x
	F	0	1	2



Der Automat ist minimal

b) $DFA = \{\{0,1,2,3\}, \{a,b,c,d\}, \delta, 0, \{0\}\}$

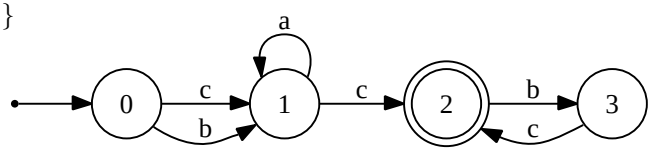
0	x			
1	x	x		
2	x	x	x	
3	x	x	x	x
	F	0	1	2



Der Automat ist minimal

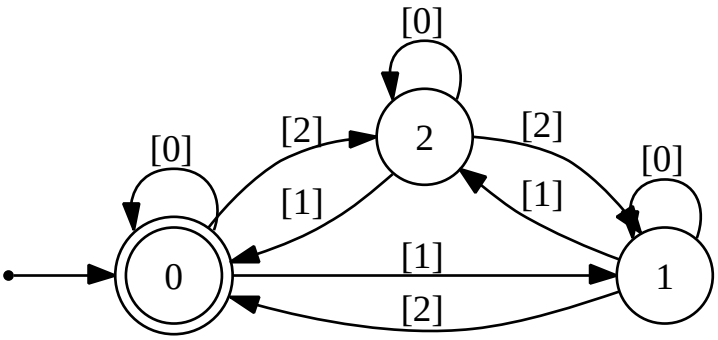
c) $DFA = \{\{0,1,2,3\}, \{a,b,c\}, \delta, 0, \{2\}\}$

0	x			
1	x	x		
2	x	x	x	
3	x	x	x	x
	F	0	1	2



Der Automat ist minimal

Aufgabe 2.
Endlicher Automat über das Alphabet $\sum = \{0, 1, \ldots, 9\}$, der die durch 3 Teilbaren Zahlen erkennt. Da eine natürliche Zahl genau dann durch 3 Teilbar ist, wenn ihre Quersumme durch 3 Teilbar ist, gibt es beim „parsen“ der Zahl genau drei Zustände: Das gelesene Teilwort geteilt durch 3 (Modulo 3) hat Rest 0, 1 oder 2. Hierbei gibt es drei Äquivalenzklassen in $\sum$, nämlich $[0] = \{0, 3, 6, 9\}$, $[1] = \{1, 4, 7\}$ und $[2] = \{2, 5, 8\}$ mit $[a] = \{x \in \sum : x \bmod 3 = a\}$. Es ist also egal, ob z.B. eine 1 oder eine 4 gelesen wird.
$DFA = \{\{0, 1, 2\}, \sum, \delta, 0, \{0\}\}$



Regulärer Ausdruck für diese Sprache:
Vereinfacht für $\sum = \{0, 1, 2\}$ (ist besser lesbar):
$\gamma = (0^*|20^*1|(20^*2|1)0^*(1(20^*1)^*(1|20^*2)|2))^*$
Vollständig:
$\gamma = ([0369]^*|[258][0369]^*[147]|([258][0369]^*[258]|[147])[0369]^*$
$([147]([258][0369]^*[147])^*([147]|[258][0369]^*[258])|[258]))^*$
Sieht etwas konfus aus, wurde mit diesem Perl–Programm ausprobiert:

```
#! /bin/perl

$n = "[0369]*";  # null (neutralelemente)
$e = "[147]";    # eins
$z = "[258]";    # zwei
$regex = "($n|$z$n$e|($z$n$z|$e)$n($e($z$n$e)*($e|$z$n$z)|$z))*";

while (1) {
        print "please enter integer: ";
        $_ = <STDIN>;
        last if(m/^$/);
        print $_/3;
        print m/^$regex$/ ? " - matched!\n" : " - failed!\n";
}
```

Aufgabe 3.

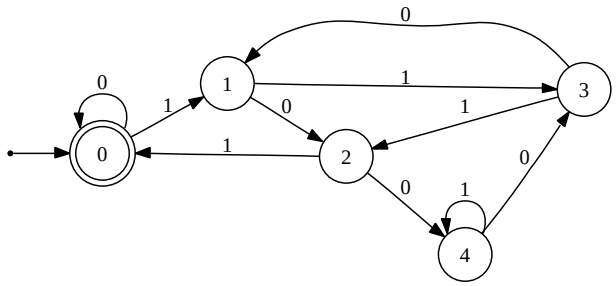
a) **Nicht regulär!** Reguläre Sprachen können nicht beliebig „zählen“. Es kann also nicht gespeichert werden, wie viele $a's$ bereits gelesen wurden, und somit auch nicht sicher gestellt werden, dass mehr bzw. gleich viele $a's$ wie $b's$ gelesen wurden. Es gibt auch einen „Pumping Lemma“ Gegenbeweis, wobei ich die Korrektheit von Pumping Lemma bis jetzt nicht wirklich nachvollziehen kann!?

b) **Nicht regulär!** Die Bedingung lässt sich umformen: $(\#_a > 0 \wedge \#_b \geq \#_c)$. Dabei ist zu sehen, dass das Teilwort b^*c^* genau die Sprache aus Teilaufgabe a) darstellt, welche nicht regulär ist. Also kann auch diese Sprache nicht regulär sein.

c) **Regulär!** Bei dieser Sprache wird nicht beliebig gezählt, wie bei a) und b), sondern bis zu einem konkreten (endlichen) Wert. Jede Primfaktorzerlegung eines Vielfachen von 5 enthält mindestens eine 5. Es muss also Entweder die Anzahl der $a's$, oder die Anzahl der $b's$ ein Vielfaches von 5 sein. Der Reguläre Ausdruck für diese Sprache lautet somit: $\gamma = ((aaaaa)^*b^*|a^*(bbbbb)^*)$

d) **Regulär!** Das Wort muss lediglich mit 0 oder 5 enden, dann ist es Element der Sprache. Regulärer Ausdruck: $\gamma = [0123456789]^*[05]$

e) **Regulär!** Wenn eine 0 gelesen wird, bedeutet das für die bereits gelesene Teilzahl: $n \xrightarrow{0} n*2$. Analog dazu mit 1: $n \xrightarrow{1} n*2+1$. Somit lassen sich 5 verschiedene Zustände ansteuern, die das Lesen eines Vielfachen von 5 möglich macht. Folgender Automat ließt die in der Sprache enthaltenen Wörter:



f) **Regulär!** a^n muss mindestens ein a enthalten, $bin(n)$ muss mindestens eine 1 enthalten. Regulärer Ausdruck: $\gamma = aa^*\$[01]^*1[01]^*$