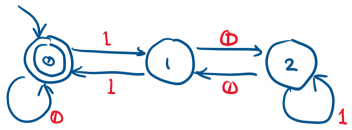
1. ***Really!?*** Prove (formally) that the language of the regular expression $\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{10}^*$ corresponding precisely to the set of positive integers divisible by 3, represented in binary (potentially with leading 0s).

   **Solution:** Let $L$ denote the language containing all binary integers divisible by 3. Construct the following DFA $M$ that recognizes language $L$, as illustrated in the figure: Denote the states of $M$ be $\{q_0, q_1, q_2\}$ as labelled, define $\delta(q_j, \texttt{0}) = q_{2j \bmod 3}$ and $\delta(q_j, \texttt{1}) = q_{(2j+1) \bmod 3}$ for any $j \in \{0, 1, 2\}$ Denote $\delta^*$ as the extended transition function of $M$. State $q_0$ is both the starting and (the only) accepting state of $M$.




   To prove that DFA $M$ indeed recognizes $L$, we will show the following statement by induction on the length of the strings:

   > An arbitrary binary string $w$ satisfies $\delta^*(q_0, w) = q_i$ if and only if $n(w)$ equals $i$ modulo 3, where $n(w)$ is the integer represented by $w$.

   Let $w$ be an arbitrary binary string.

   - If $w = \varepsilon$, $n(w) = 0$ and the machine $M$ correctly halts at $q_0$ after reading $w$.
   - If $w = x \cdot \texttt{0}$ for some string $x$ shorter than $w$, by induction $\delta^*(q_0, x) = q_j$ if and only if $n(x)$ equals $j$ modulo 3. Now $n(w)$ is equal to $2n(x)$; one can readily verify that $\delta^*(q_0, w)$ ends at $q_{n(w) \bmod 3}$ because reading a 0 from state $q_j$ results in

   $$\delta(q_j, \texttt{0}) = q_{2j \bmod 3} = q_{2n(x) \bmod 3} = q_{n(w) \bmod 3}.$$

   - If $w = x \cdot \texttt{1}$ for some string $x$ shorter than $w$, by induction $\delta^*(q_0, x) = q_j$ if and only if $n(x)$ equals $j$ modulo 3. Now $n(w)$ is equal to $2n(x) + 1$; one can readily verify that $\delta^*(q_0, w)$ ends at $q_{n(w) \bmod 3}$ because reading a 1 from state $q_j$ results in

   $$\delta(q_j, \texttt{1}) = q_{(2j+1) \bmod 3} = q_{(2n(x)+1) \bmod 3} = q_{n(w) \bmod 3}.$$

   Now we prove the following statements by induction on the length of the strings:

   - Every positive binary integer equals to 0 mod 3 has the expression $\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{10}^*$
   - Every positive binary integer equals to 1 mod 3 has the expression $\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*$
   - Every positive binary integer equals to 2 mod 3 has the expression $\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{01}^*$

   Let $w$ be an arbitrary binary string, representing a binary positive integer (possibly with leading 0s); in particular, $w$ must be nonempty and contains at least a 1.

   - *Case* $n(w) = 0$ mod 3*:* By the DFA $M$ constructed, $\delta^*(q_0, w) = q_0$ holds.
     - If $w = x \cdot \texttt{0}$, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_0$ holds. By induction statements, $x$ has the expression $\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{10}^*$. Appending 0, string $w$ has the expression

     $$\texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{10}^*\texttt{0} \subseteq \texttt{0}^*\texttt{1}(\texttt{10}^*\texttt{1} + \texttt{01}^*\texttt{0})^*\texttt{10}^*$$

     so the induction statement holds for $w$.

  - If $w = x \cdot \mathtt{1}$, if $x$ contains only $\mathtt{0}$s then $w$ has the expression $\mathtt{0}^*\mathtt{1}$ and the statement holds. Otherwise, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_1$ holds. By induction statements, $x$ has the expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*$. Appending $\mathtt{1}$, string $w$ has the expression

$$\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{1} \subseteq \mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{10}^*$$

    so the induction statement holds for $w$.

- *Case* $n(w) = 1 \bmod 3$*:* By the DFA $M$ constructed, $\delta^*(q_0, w) = q_1$ holds.
  - If $w = x \cdot \mathtt{0}$, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_2$ holds. By induction statements, $x$ has the expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*$. Appending $\mathtt{0}$, string $w$ has the expression

$$\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*\mathtt{0} \subseteq \mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*$$

    so the induction statement holds for $w$.
  - If $w = x \cdot \mathtt{1}$, if $x$ contains only $\mathtt{0}$s then $w$ has the expression $\mathtt{0}^*\mathtt{1}$ and the statement holds. Otherwise, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_0$ holds. By induction statements, $x$ has the expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{10}^*$. Appending $\mathtt{1}$, string $w$ has the expression

$$\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{10}^*\mathtt{1} \subseteq \mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*$$

    so the induction statement holds for $w$.

- *Case* $n(w) = 2 \bmod 3$*:* By the DFA $M$ constructed, $\delta^*(q_0, w) = q_2$ holds.
  - If $w = x \cdot \mathtt{0}$, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_1$ holds. By induction statements, $x$ has the expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*$. Appending $\mathtt{0}$, string $w$ has the expression

$$\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{0} \subseteq \mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*$$

    so the induction statement holds for $w$.
  - If $w = x \cdot \mathtt{1}$, if $x$ contains only $\mathtt{0}$s then $w$ has the expression $\mathtt{0}^*\mathtt{1}$ and the statement holds. Otherwise, string $x$ must be associated with another positive integer. Also, $\delta^*(q_0, x) = q_2$ holds. By induction statements, $x$ has the expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*$. Appending $\mathtt{1}$, string $w$ has the expression

$$\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*\mathtt{1} \subseteq \mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{01}^*$$

    so the induction statement holds for $w$.

In all six cases, the induction statements hold for any arbitrary binary string $w$; this concludes the induction proof. As a direct consequence, the set of positive binary integers divisible by 3 corresponds exactly to the regular expression $\mathtt{0}^*\mathtt{1}(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*\mathtt{10}^*$. (The forward inclusion is proved by the above induction, and the backward inclusion can be readily verified by running the machine on an arbitrary string of the said expression: the $\mathtt{0}^*\mathtt{1}$ part takes us to state $q_1$, any string in $(\mathtt{10}^*\mathtt{1} + \mathtt{01}^*\mathtt{0})^*$ makes the DFA stays at $q_1$, and the final $\mathtt{10}^*$ takes us back to state $q_0$ and thus the DFA accepts.) ■

**Rubric:** Some of you might attempt to prove the equivalence directly between positive binary integers and the (strengthened) regular expressions, which is also valid. The difficult direction is always to prove that the strings can be represented by the said regular expressions. Because we did the proof that the language DFA $M$ is precisely the set of all binary integers divisible by 3 during the string induction lecture, you can use the fact directly in your answer and no points will be taken from your proof attempt (if you included one).

Standard 5-point grading scale. Maximum 3 point if no justification whatsoever for the "regular expression $\rightarrow$ multiples of 3" direction.

- 5 means the answer is perfect — the solution is correct, concise, and convincing.
- 4 means the answer is basically correct, modulo some small minor issues.
- 3 means the answer is mostly correct, but there are some missing details or errors that does not affect the overall argument.
- 2 means the answer is heading in the right direction, but there are major gaps in the presentation before making the solution rigorous.
- 1 means the answer is incorrect, despite an honest attempt. The solution successfully convey the (while incorrect) ideas to the reader/grader.
- 0 means the answer is either missing, for the wrong question, or not readable/parsable.

Proofs using *backward induction* and *proof by examples* will receive an automatic ***zero***. Proofs containing *undefined variables*, *unparsable sentences* and *proof sketches* will trigger ***sudden-death***; you will receive a zero unless the proof is otherwise *perfect*. It is up to the graders to decide if they can understand your proof or not.

2. ***Erasing digit sequence.*** Let the input be a string of digits from 0 to 9 (in other words, the alphabet set $\Sigma$ is $\{0,\ldots,9\}$). The ERASE function is defined as follows:

```
ERASE(w):
  input: digit string w
  digit string r ← ε
  while w is not empty:
      d ← first digit of w
      remove the first digit of w
      r ← r · d ⟨⟨append d after r⟩⟩
      if there are at least d digits left in w:
          remove d digits from w
      else:
          return fail
  return r
```

A digit string $w$ is ***erasable*** if ERASE$(w)$ successfully returns another digit string. For example, string $w = 31415926535897932384626433832795028841 9$ is erasable because

$$\text{ERASE}(w) = \cancel{3}1415926535\cancel{8}979323\cancel{8}4626\cancel{4}33\cancel{8}327\cancel{9}50\cancel{2}88\cancel{4}\cancel{1}9 = 355243251.$$

Construct DFAs that recognize the following languages.

(a) $\{w \in \Sigma^* : w \text{ is erasable}\}$

**Solution:** First let's construct an *eraser gadget* recognizing the language

$$L_0 = \{d \cdot \Sigma^d \in \Sigma^* : d \text{ is a digit from 0 to 9}\}.$$

Define a ***fuse*** of length $d$ to be a directed path containing nodes $q_d,\ldots,q_0$ and $d$ edges $q_i \to q_{i-1}$ for each $i \in 1,\ldots,d$. We say the fuse starts from state $q_d$ and ends at $q_0$. One should think of the nodes as *states* and directed edges as *transitions*. We associate each transition with all the digits in $\Sigma$; in other words, $\delta(q_i, a) = q_{i-1}$ for any digits $i$ and $a$.

Construct the ***eraser gadget $G(s,t)$*** by taking a fuse of length $d$ for each digit $d$, and merge all the ends together into a single state $t$. Add an extra starting state $s$ and create transitions from $s$ to the head of length-$d$ fuse on reading digit $d$ for any digit $d$.

The gadget $G(s,t)$ can be viewed as a DFA if we set $t$ to be an accepting state and add a *failed state* $f$ and the corresponding transitions $\delta(t,a) = f$ and $\delta(f,a) = f$ for any digit $a$. Such DFA recognize the language $L_0$: Let $w$ be an arbitrary digit sequence. On reading the first digit $d$, $G(s,t)$ moves to the head of the length-$d$ fuse. The machine reads $d$ more digits using the length-$d$ fuse and accepts if we are at the end of the string. If there are more or less than $d$ digits in $w$ following the initial digit then the machine correctly rejects.

Now we use the eraser gadget $G(s,t)$ to build DFA $M_1$ recognizing the language in the statement: merge the two states $s$ and $t$ into a single state. DFA $M_1$ iteratively reads chunk of digits in $L_0$ (each of the form $d \cdot \Sigma^d$ for some digit $d$); $M_1$ accepts $w$ if and only if string $w$ can be decomposed into (possibly zero) concatenation of such chunks, which is equivalent to saying that $w$ is erasable. ■

(b) $\{w \in \Sigma^* : \text{both } w \text{ and ERASE}(w) \text{ are erasable}\}$

**Solution:** We use the fuses, the eraser gadget, and DFA $M_1$ from the previous question to build DFA $M_2$ that recognizes the language in the statement.

Let $M_1$ be the DFA constructed in problem 2(a). For each digit $d$, replace the transition (directed edge) from $s$ to the head of the length-$d$ fuse $h_d$ by inserting another length-$d$ fuse, identifying the end of the inserted fuse with $h_d$, and adding a transition from $s$ to the head of

the inserted fuse with edge label $d$. Replace each of the transitions in the original length-$d$ fuse with an eraser gadget. The newly constructed machine $M_2$ is a DFA with the same starting and ending state $s$ as $M_1$, and each state has exactly 10 outgoing transitions, each corresponds to a different digit in $\Sigma$.

DFA $M_2$ iteratively reads chunk of digits of the form $d \cdot \Sigma^d \cdot L_0^d$. DFA $M_2$ accepts $w$ if and only if string $w$ can be decomposed into (possibly zero) concatenation of chunks of strings in $d \cdot \Sigma^d \cdot L_0^d$ (corresponding to $\text{ERASE}(w)$ being erasable), where each chunk can be further decomposed into (possibly zero) concatenation of chunks of the form $d \cdot \Sigma^d$ (corresponding to $w$ being erasable). ■

**Note.** There are more efficient ways to construct DFAs recognizing the same languages, with much fewer states (like many of you did in your submissions). However, I hope you see the benefit in designing *modules/gadgets* separately and reusing them again and again throughout the construction. This is equivalent to using functions and libraries in your program. While we are now working with extremely low-level programming languages, it is important to remember that our eventual goal is to model arbitrary computation using one universal model; trusting that basic functions can be implemented and thus using them freely in your high-level algorithm design is something we want you to be comfortable with.

**Rubric:** No formal proofs for the correctness are required, but your ***must*** include a brief explanation of why your DFA works. Maximum 3 points if the machine constructed is an NFA instead (say with multiple transitions from a state with the same digit label).

Standard 5-point grading scale ***for each subproblem*** plus deadly-sins and sudden-death rules. Constructions without English explanations receive an automatic ***zero***.