1. Prove whether the following languages are regular or not.

   (a) Strings over the alphabet $\Sigma = \{0, \ldots, 9, \#\}$ that contain a substring $c\#^c c$, where $c \in \{0, \ldots, 9\}$. E.g., 382103###38592, 7892##234 and 00 are in the language.

   (b) Strings over the alphabet $\Sigma = \{0, \ldots, 9, \#\}$ of the form $\langle n \rangle \#^n$, where n is a sequence of digits interpreted as a decimal number. E.g., 0, 3###, 11########### are in the language.

   (c) Strings over the alphabet $\Sigma = \{a, b, \ldots, z\}$ that have the same 3 characters repeated in two places. E.g., ur**ban**e**ban**ana, trampo**lin**ejugg**lin**g, acclim**ati**z**ati**on.

2. Let $f : \Sigma_1 \to \Sigma_2^*$ be a function from symbols in one alphabet to strings in another. We can extend f to apply to strings in Σ_1^* by the following recursive definition:

$$f(\epsilon) = \epsilon$$
$$f(ax) = f(a) \cdot f(x) \qquad \text{for } a \in \Sigma_1, x \in \Sigma_1^*$$

   Likewise, we can apply f to languages by defining $f(L) = \{f(w) | w \in L\}$.

   f is known as a *language homomorphism*. For example, we can define f to map `0` to `batman` and `1` to `robin`, then $f(110) =$ `robinrobinbatman`. As another example, we can define f_{ASCII} that maps each character to its 8-bit ASCII binary representation, in which case $f_{\text{ASCII}}($`374`$) =$ `001100110011011100110100`.

   Given a DFA M that accepts L, show how to construct an NFA N that accepts $f(L)$. Formally prove the correctness of your construction.

   Note that we are looking for an explicit construction of an NFA here, rather than simply a proof that $f(L)$ is regular, which implies the existence of such an NFA N.

3. Give a context-free grammar for the following languages. You must specify what language is generated by each non-terminal and briefly explain why.

   (a) Binary strings that have remainder of 2 when divided by 5 (e.g., `111`, `10`, `10001`).

   (b) Strings over the alphabet $\{$`0`, `1`$\}$ that have two blocks of 0's of equal length. E.g., `001100010001110` or `10110011100010` but not `0` or `0100`.

   (c) Arithmetic expressions over decimal numbers using addition (`+`), multiplication (`*`), and exponentiation (`^`) with minimal parentheses. Here are the rules:

   - The usual precedence rules apply, so `1+2*3^4` is equivalent to `1+(2*(3^4))`
   - Any parentheses that could be removed without changing the meaning of the expression are not allowed. E.g., `1+(2*(3^4))` is an invalid expression, as are `(2*3)+5`, `3+(4+8)`, `(4+6)`, `3^((4+5))`. `2*(3+5)`, however, is valid.
   - Since exponentiation is not associative, any double (or more) exponentiation must be parenthesized to remove ambiguity. I.e., `2^3^4` is invalid, instead you have to write `(2^3)^4` or `2^(3^4)`. Likewise `(1+2)^(3*4)^5` is invalid.

Solved problem

4. Let L be the set of all strings over $\{$`0`, `1`$\}^*$ with exactly twice as many `0`s as `1`s.

   (a) Describe a CFG for the language L.

   [Hint: For any string u define $\Delta(u) = \#(\texttt{0}, u) - 2\#(\texttt{1}, u)$. Introduce intermediate variables that derive strings with $\Delta(u) = 1$ and $\Delta(u) = -1$ and use them to define a non-terminal that generates L.]

   Solution: $S \to \varepsilon \mid SS \mid \texttt{00}S\texttt{1} \mid \texttt{0}S\texttt{1}S\texttt{0} \mid \texttt{1}S\texttt{00}$ ■

(b) Prove that your grammar G is correct. As usual, you need to prove both $L \subseteq L(G)$ and $L(G) \subseteq L$. *[Hint: Let $u_{\le i}$ denote the prefix of u of length i. If $\Delta(u) = 1$, what can you say about the smallest i for which $\Delta(u_{\le i}) = 1$? How does u split up at that position? If $\Delta(u) = -1$, what can you say about the smallest i such that $\Delta(u_{\le i}) = -1$?]*

Solution: We separately prove $L \subseteq L(G)$ and $L(G) \subseteq L$ as follows:

Claim 1. *$L(G) \subseteq L$, that is, every string in $L(G)$ has exactly twice as many* `0`*s as* `1`*s.*

Proof: As suggested by the hint, for any string u, let $\Delta(u) = \#(\mathtt{0}, u) - 2\#(\mathtt{1}, u)$. We need to prove that $\Delta(w) = 0$ for every string $w \in L(G)$.

Let w be an arbitrary string in $L(G)$, and consider an arbitrary derivation of w of length k. Assume that $\Delta(x) = 0$ for every string $x \in L(G)$ that can be derived with fewer than k productions.[1] There are five cases to consider, depending on the first production in the derivation of w.

- If $w = \varepsilon$, then $\#(\mathtt{0}, w) = \#(\mathtt{1}, w) = 0$ by definition, so $\Delta(w) = 0$.
- Suppose the derivation begins $S \rightsquigarrow SS \rightsquigarrow^* w$. Then $w = xy$ for some strings $x, y \in L(G)$, each of which can be derived with fewer than k productions. The inductive hypothesis implies $\Delta(x) = \Delta(y) = 0$. It immediately follows that $\Delta(w) = 0$.[2]
- Suppose the derivation begins $S \rightsquigarrow \mathtt{00}S\mathtt{1} \rightsquigarrow^* w$. Then $w = \mathtt{00}x\mathtt{1}$ for some string $x \in L(G)$. The inductive hypothesis implies $\Delta(x) = 0$. It immediately follows that $\Delta(w) = 0$.
- Suppose the derivation begins $S \rightsquigarrow \mathtt{1}S\mathtt{00} \rightsquigarrow^* w$. Then $w = \mathtt{1}x\mathtt{00}$ for some string $x \in L(G)$. The inductive hypothesis implies $\Delta(x) = 0$. It immediately follows that $\Delta(w) = 0$.
- Suppose the derivation begins $S \rightsquigarrow \mathtt{0}S\mathtt{1}S\mathtt{1} \rightsquigarrow^* w$. Then $w = \mathtt{0}x\mathtt{1}y\mathtt{0}$ for some strings $x, y \in L(G)$. The inductive hypothesis implies $\Delta(x) = \Delta(y) = 0$. It immediately follows that $\Delta(w) = 0$.

In all cases, we conclude that $\Delta(w) = 0$, as required. □

Claim 2. *$L \subseteq L(G)$; that is, G generates every binary string with exactly twice as many* `0`*s as* `1`*s.*

Proof: As suggested by the hint, for any string u, let $\Delta(u) = \#(\mathtt{0}, u) - 2\#(\mathtt{1}, u)$. For any string u and any integer $0 \le i \le |u|$, let $\boldsymbol{u_i}$ denote the ith symbol in u, and let $\boldsymbol{u_{\le i}}$ denote the prefix of u of length i.

Let w be an arbitrary binary string with twice as many `0`s as `1`s. Assume that G generates every binary string x that is shorter than w and has twice as many `0`s as `1`s. There are two cases to consider:

- If $w = \varepsilon$, then $\varepsilon \in L(G)$ because of the production $S \to \varepsilon$.
- Suppose w is non-empty. To simplify notation, let $\Delta_i = \Delta(w_{\le i})$ for every index i, and observe that $\Delta_0 = \Delta_{|w|} = 0$. There are several subcases to consider:
  - Suppose $\Delta_i = 0$ for some index $0 < i < |w|$. Then we can write $w = xy$, where x and y are non-empty strings with $\Delta(x) = \Delta(y) = 0$. The induction hypothesis implies that $x, y \in L(G)$, and thus the production rule $S \to SS$ implies that $w \in L(G)$.
  - Suppose $\Delta_i > 0$ for all $0 < i < |w|$. Then w must begin with `00`, since otherwise $\Delta_1 = -2$ or $\Delta_2 = -1$, and the last symbol in w must be `1`, since otherwise $\Delta_{|w|-1} = -1$. Thus, we can write $w = \mathtt{00}x\mathtt{1}$ for some binary string x. We easily observe that $\Delta(x) = 0$, so the induction hypothesis implies $x \in L(G)$, and thus the production rule $S \to \mathtt{00}S\mathtt{1}$ implies $w \in L(G)$.

[1] Alternatively: Consider the *shortest* derivation of w, and assume $\Delta(x) = 0$ for every string $x \in L(G)$ such that $|x| < |w|$.

[2] Alternatively: Suppose the *shortest* derivation of w begins $S \rightsquigarrow SS \rightsquigarrow^* w$. Then $w = xy$ for some strings $x, y \in L(G)$. Neither x or y can be empty, because otherwise we could shorten the derivation of w. Thus, x and y are both shorter than w, so the induction hypothesis implies.... We need some way to deal with the decompositions $w = \varepsilon \bullet w$ and $w = w \bullet \varepsilon$, which are both consistent with the production $S \to SS$, without falling into an infinite loop.

- Suppose $\Delta_i < 0$ for all $0 < i < |w|$. A symmetric argument to the previous case implies $w = \mathtt{1}x\mathtt{00}$ for some binary string x with $\Delta(x) = 0$. The induction hypothesis implies $x \in L(G)$, and thus the production rule $S \to \mathtt{1}S\mathtt{00}$ implies $w \in L(G)$.
- Finally, suppose none of the previous cases applies: $\Delta_i < 0$ and $\Delta_j > 0$ for some indices i and j, but $\Delta_i \neq 0$ for all $0 < i < |w|$.

  Let i be the smallest index such that $\Delta_i < 0$. Because Δ_j either increases by 1 or decreases by 2 when we increment j, for all indices $0 < j < |w|$, we must have $\Delta_j > 0$ if $j < i$ and $\Delta_j < 0$ if $j \geq i$.

  In other words, there is a *unique* index i such that $\Delta_{i-1} > 0$ and $\Delta_i < 0$. In particular, we have $\Delta_1 > 0$ and $\Delta_{|w|-1} < 0$. Thus, we can write $w = \mathtt{0}x\mathtt{1}y\mathtt{0}$ for some binary strings x and y, where $|\mathtt{0}x\mathtt{1}| = i$.

  We easily observe that $\Delta(x) = \Delta(y) = 0$, so the inductive hypothesis implies $x, y \in L(G)$, and thus the production rule $S \to \mathtt{0}S\mathtt{1}S\mathtt{0}$ implies $w \in L(G)$.

In all cases, we conclude that G generates w. □

Together, Claim 1 and Claim 2 imply $L = L(G)$. ■

Rubric: 10 points:
- part (a) = 4 points. As usual, this is not the only correct grammar.
- part (b) = 6 points = 3 points for $\subseteq$ + 3 points for $\supseteq$, each using the standard induction template (scaled).