0. Conceptual Review
(a) Regular expression rules:
Basis: $\epsilon$, $a$ for $a \in \Sigma$
Recursive: If $A, B$ are regular expressions, $(A \cup B), AB$, and $A^*$ are regular expressions.

1. Regular Expressions Warmup
Consider the following Regular Expression (RegEx):

$$1(45 \cup 54)^*1$$

List 5 strings accepted by the RegEx and 5 strings from $T := \{1, 4, 5\}^*$ rejected by the RegEx. Then, summarize this RegEx in your own words.

Solution:

Accepted: 
- 1451
- 1541
- 145541
- 1454545451
- 11

Rejected: 
- 1
- 1441
- 45
- 14451
- 111

This RegEx accepts exactly those strings that start and end with a 1, and have zero or more pairs of 45 or 54 in the middle.
2. Context Free Grammars Warmup
Consider the following CFG which generates strings from the language $V := \{0, 1, 2, 3, 4\}^*$

$$
S \rightarrow 0X4 \\
X \rightarrow 1X3 \mid 2
$$

List 5 strings generated by the CFG and 5 strings from $V$ not generated by the CFG. Then, summarize this CFG in your own words.

**Solution:**

**Accepted:**
- 024
- 01234
- 0112334
- 011123334
- 01111233334

**Rejected:**
- $\epsilon$
- 2
- 0244
- 011234
- 10234

This CFG is all strings of the form $0 \ 1^m \ 2 \ 3^m \ 4$, where $m \geq 0$. That is, it’s all strings made of one 0, followed by zero or more 1’s, followed by a 2, followed by the same number of 3’s as 1’s, followed by one 4.

3. Simplify the RegEx
Consider the following Regular Expression (RegEx):

$$
0^* (0 \cup 1)^* ((01) \cup (11) \cup (10) \cup (00)) 1^* (0 \cup 1)^*
$$

List 3 strings accepted by the RegEx and 3 strings from $S := \{0, 1\}^*$ rejected by the RegEx. Then, summarize this RegEx in your own words and write a simpler RegEx that accepts exactly the same set of strings.

**Solution:**

**Accepted:**
- 01
- 10
- 10100100101

**Rejected:**
- $\epsilon$
- 0
- 1

This RegEx accepts all binary strings that are 2 or more characters long. A simpler RegEx for this is $(0 \cup 1)(0 \cup 1)(0 \cup 1)^*$. 
4. Constructing RegExs and CFGs

For each of the following, construct a regular expression and CFG for the specified language.

(a) Strings from the language $S := \{a\}^*$ with an even number of $a$'s.

Solution:

$$\begin{align*}
(aa)^* \\
S &\rightarrow aaS|\varepsilon
\end{align*}$$

(b) Strings from the language $S := \{a, b\}^*$ with an even number of $a$'s.

Solution:

$$\begin{align*}
b^*(b^*ab^*ab^*)^* \\
S &\rightarrow bS|aSaS|\varepsilon
\end{align*}$$

(c) Strings from the language $S := \{a, b\}^*$ with odd length.

Solution:

$$\begin{align*}
(aa \cup ab \cup ba \cup bb)^*(a \cup b) \\
S &\rightarrow CS\varepsilon \\
C &\rightarrow aa\varepsilon | abC | baC | bbC | \varepsilon
\end{align*}$$

(d) (Challenge) Strings from the language $S := \{a, b\}^*$ with an even number of $a$'s or an odd number of $b$'s.

Solution:

$$\begin{align*}
b^*(b^*ab^*ab^*)^* \cup (a^* \cup a^*ba^*ba^*)^*b(a^* \cup a^*ba^*ba^*)^* \\
S &\rightarrow E|bO \\
E &\rightarrow EE|aEa|b|\varepsilon \\
O &\rightarrow OO|bO|a|\varepsilon
\end{align*}$$
5. Structural Induction: CFGs
Consider the following CFG:

\[ S \rightarrow SS \mid 0S1 \mid 1S0 \mid \epsilon \]

Prove that every string generated by this CFG has an equal number of 1’s and 0’s.

**Hint 1:** Start by converting this CFG to a recursively defined set.

**Hint 2:** You may wish to define the functions \( \#_0(x), \#_1(x) \) on a string \( x \).

**Solution:**
First we observe that the language defined by this CFG can be represented by a recursively defined set. Define a set \( S \) as follows:

- **Basis Rule:** \( \epsilon \in S \)
- **Recursive Rule:** If \( x, y \in S \), then \( 0x1, 1x0, xy \in S \).

Now we perform structural induction on the recursively defined set. Define the functions \( \#_0(t), \#_1(t) \) to be the number of 0’s and 1’s respectively in the string \( t \).

**Proof.** For a string \( t \), let \( P(t) \) be defined as "\( \#_0(t) = \#_1(t) \)". We will prove \( P(t) \) is true for all strings \( t \in S \) by structural induction.

**Base Case** \((t = \epsilon)\): By definition, the empty string contains no characters, so \( \#_0(\epsilon) = 0 = \#_1(\epsilon) \)

**Inductive Hypothesis:** Suppose \( P(x) \) and \( P(y) \) hold for arbitrary strings \( x, y \in S \).

**Inductive Step:**
**Case 1:** Goal: show \( P(0x1) \).
By the IH, \( \#_0(x) = \#_1(x) \). Then observe that:

\[ \#_0(0x1) = \#_0(x) + 1 = \#_1(x) + 1 = \#_1(0x1) \]

Therefore \( \#_0(0x1) = \#_1(0x1) \). This proves \( P(0x1) \).

**Case 2:** Goal: show \( P(1x0) \)
By the IH, \( \#_0(x) = \#_1(x) \). Then observe that:

\[ \#_0(1x0) = \#_0(x) + 1 = \#_1(x) + 1 = \#_1(1x0) \]

Therefore \( \#_0(1x0) = \#_1(1x0) \). This proves \( P(1x0) \).

**Case 3:** Goal: show \( P(xy) \)
By the IH, \( \#_0(x) = \#_1(x) \) and \( \#_0(y) = \#_1(y) \). Then observe that:

\[ \#_0(xy) = \#_0(x) + \#_0(y) = \#_1(x) + \#_1(y) = \#_1(xy) \]

Therefore \( \#_0(xy) = \#_1(xy) \). This proves \( P(xy) \).
So by structural induction, \( P(t) \) is true for all strings \( t \in S \). \( \square \)

Since the recursively defined set, \( S \), is exactly the set of strings generated by the CFG, we have proved that the statement is true for every string generated by the CFG too.