No books, no calculators. One 8.5x11 page of handwritten notes.

Name: ____________________________

Student ID:_______________________

ieng6.ucsd.edu login: ________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/15</td>
</tr>
<tr>
<td>2</td>
<td>/15</td>
</tr>
<tr>
<td>3</td>
<td>/15</td>
</tr>
<tr>
<td>4</td>
<td>/15</td>
</tr>
<tr>
<td>5</td>
<td>/15</td>
</tr>
<tr>
<td>6</td>
<td>/15</td>
</tr>
<tr>
<td>Total</td>
<td>/90</td>
</tr>
</tbody>
</table>
1. 15 pts. Determine whether regular languages are closed under the following operation (that is, if \(L\) is a regular language, is \(\text{Double}(L)\) necessarily a regular language?). Prove your answer.

- \(\text{Double}(L) = \{x|x = vw, v \in L, w \in L, \text{ and } |v| = |w|\}\)

  **Solution:** Not regular.

  We’ll use a counter-example. Let \(L = 1^*0\). Then \(\text{Double}(L) = 1^n0^10^n\).

  Assume for the sake of contradiction that \(\text{Double}(L)\) is regular. Let \(p\) be as in the pumping lemma. Then \(1^p01^p0 \in \text{Double}(L)\).

  But, by the pumping lemma, that means there exists a \(x, y, z\) such that \(|xy| \leq p, |y| > 0, \text{ and } \forall i, xy^iz \in \text{Double}(L)\). But since \(|xy| \leq p\), that means \(y\) is completely contained within the first 1’s. So, letting \(i = 0, xy^iz = xz \notin \text{Double}(L)\) since it’s of the form \(1^q01^p0\) where \(q < p\). Thus, our assumption that \(\text{Double}(L)\) is regular is incorrect.
2. 15 pts. Convert the following nondeterministic finite automaton to an equivalent deterministic one.

![Nondeterministic Finite Automaton Diagram]

Solution: Using the construction where we start with the start state (and all states reachable from there), and then see what set of states we can end up in on each symbol, we end up with:
3. 15 pts. Given the language \( L = \{0^n1^n2^m|n,m \geq 0\} \), prove that \( L \) is not context-free, or give a pushdown automata that recognizes it.

Solution: We’ll push an \( X \) symbol when we see a 0, and then pop it off when we see a 1. If we reach the bottom of the stack while reading 0’s, then we start pushing \( Y \) symbols (we use non-determinism to guess when to pop \( X \)’s or when to push \( Y \)’s). When reading 2’s we match them to \( Y \) symbols. If the stack is empty when we reach the end of the string, we accept. We must be careful to deal with empty 0’s, 1’s, and 2’s.
4. 15 pts. Given the language \( L = \{0^n1^m2^n1^m | m, n \geq 0\} \), prove that \( L \) is not context-free or give a context-free grammar that generates it.

Solution:

Assume for the sake of contradiction that \( L \) is context-free. Let \( p \) be as in the pumping lemma. Let \( s = 0^p1^p2^p1^p \). Since \( s \in L \) and \(|s| \geq p\), by the pumping lemma, there exists a \( u, v, x, y, z \) such that \(|vxy| \leq p\), \(|vy| > 0\), and \( \forall i, uv^ixy^iz \in L \).

If \( vxy \) is completely contained within one run of symbols, then \( s' = uxz \) will have too few of that symbol, and so \( s' \notin L \).

The substring \( vxy \) spans at most two adjacent symbol types. Without loss of generality, assume it’s the middle span: 1’s and 2’s. Letting \( i = 0 \), \( s' = uv^0xy^0z = uxz \) will contain fewer 1s at the front than 1s at the end, or fewer 2’s than 0s. Thus, \( s' \notin L \) and therefore our assumption that \( L \) is context-free is incorrect.
5. 15 pts. Begin the process of converting the following grammar to Chomsky Normal Form by adding a new start nonterminal and removing $\varepsilon$-productions (do not perform the remaining steps):

$$
S \rightarrow ASB | AB \\
A \rightarrow aS | a | \varepsilon \\
B \rightarrow SS | A | bb
$$

Solution: To begin with, add a new start variable:

$$
S' \rightarrow S' \\
S \rightarrow ASB | AB \\
A \rightarrow aS | a | \varepsilon \\
B \rightarrow SS | A | bb
$$

Remove epsilon productions. First, remove $A \rightarrow \varepsilon$:

$$
S' \rightarrow S' \\
S \rightarrow SB | ASB | B | AB \\
A \rightarrow aS | a \\
B \rightarrow SS | A | \varepsilon | bb
$$

Next, remove $B \rightarrow \varepsilon$:

$$
S' \rightarrow S' \\
S \rightarrow SB | ASB | B | AB \\
A \rightarrow aS | a \\
B \rightarrow SS | A | bb
$$

Finally, remove $S \rightarrow \varepsilon$:

$$
S' \rightarrow S' \varepsilon \\
S \rightarrow S | B | SB | A | AS | AB | ASB | A | AB \\
A \rightarrow a | aS | a \\
B \rightarrow S | SS | A | bb
$$
Remove duplicate productions:

\[ S' \rightarrow S|\epsilon \]
\[ S \rightarrow S|B|SB|A|AS|AB|ASB \]
\[ A \rightarrow a|aS \]
\[ B \rightarrow S|SS|A|bb \]
6. 15 pts. Convert the following context-free grammar to a pushdown automaton (using the general construction given in the book or in class):

\[
\begin{align*}
S & \rightarrow AB | BC | AC | \varepsilon \\
A & \rightarrow AA | CC | a | b \\
B & \rightarrow BA | b \\
C & \rightarrow CC | c
\end{align*}
\]

Note: In your pushdown automaton, you may use the shortcut notation \(a, b \rightarrow cde\) which represents popping the symbol \(b\), then pushing \(e\), then \(d\), then \(c\).

Solution: Since this grammar is in Chomsky-Normal Form, we can use the CNF construction. We add an \(\varepsilon\)-transition for every production that doesn’t lead to a terminal, and we consume input for every production that leads to a terminal:
Alternatively, one can use the construction that doesn’t require CNF: add an \( \varepsilon \)-transition for every production, and then add transitions for each terminal symbol: