Last 4 digits of your UTEP ID number:

General comments:

- you are allowed up to 5 pages of handwritten notes;
- if you need extra pages, place the last 4 digits of ID number on each extra page;
- the main goal of most questions is to show that you know the corresponding algorithms; so, if you are running out of time, just follow the few first steps of the corresponding algorithm;
- each question will be graded on its own merit; so, for example, if when answering to the first part of the question, you got a wrong grammar, but on the second part, you correctly traced the new grammar, you will get full credit for the second part.

Good luck!

1. A student who gets only As and Bs is classified as cum laude during the commencement when his or her GPA is 3.5 is above, i.e., if he/she has at least as many As as Bs. For example, students with records ABAB or ABA qualify. Prove that the language consisting of all sequences of As and Bs that make the student qualified is not regular.

Theorem: \( L = \{ \text{any sequence that has an equal number of As and Bs} \} \) is not regular. Proof by contradiction: Let's assume that \( L \) is regular. Then by pumping lemma, there exists a number \( p \) such that every word \( w \) from \( L \) whose length is at least \( p \) can be represented as \( xy^iz \), where \( \text{len}(y) > 0, \text{len}(xy) \leq p \) and for every \( i, xy^iz \) is in \( L \).

Let's take the word \( w = B \ldots B \ A \ldots A \), It's length is \( 2p \geq p \) so \( w \) can be represented as \( xy^iz \) in accordance with the pumping lemma. The word \( w=xy^iz \) starts with \( xy \) and \( \text{len}(xy) \leq p \) so \( x \) and \( y \) are among the first \( p \) symbols of the word, \( xy \) consists of only Bs, but we don't add As. So, in \( xy^iz \), we have an more Bs than As, thus, \( y^iz \) is not in \( L \). But by pumping lemma \( xyyz \) is in \( L \), we get a contradiction, so our assumption is wrong and \( L \) is not regular.
2. Design a finite automaton for recognizing binary sequences that have even number of 0s and even number of 1s. Assume that the input strings contain only symbols 0 and 1. The easiest is to have 4 states:

- the desired (final) state ee in which we have even number of 0s and even number of 1s,
- the state eo, in which we have even number of 0s and odd number of 1s, and
- similarly defined states oe (odd-even) and oo (odd-odd).

You just need to describe transitions between these states, and which states are final. Show, step-by-step, how your automaton will accept the string 1010.
3. Use a general algorithm to transform the finite automaton from Problem 2 into a context-free grammar (CFG). Show, step-by-step, how this CFG will generate the word 1010.
4-5. Use a general algorithm to translate the CFG from Problem 3 into an appropriate push-down automaton. Explain, step-by-step, how this automaton will accept the word 1010.

```
Rules
ee → ε
eo → leo
ee → Ooe
eo → Oeo
oc → Oeo
oe → l0e
0c → l0c
```

![Diagram of push-down automaton]

**Tracing**

```
Step 1: ε, ε → ε
Step 2: ε → eo
Step 3: eo → eo
Step 4: eo → eo
Step 5: eo → eo
Step 6: eo → eo
Step 7: eo → eo
Step 8: eo → eo
Step 9: eo → eo
Step 10: eo → eo
Step 11: ε

file:///Q:/cs3350.19/test2.html

3/9/2019
```
6-7. Use a general algorithm to translate the CFG from Problem 3 into Chomsky normal form.

Rules:

- $e ightarrow e$
- $e ightarrow le$
- $e ightarrow lce$
- $e ightarrow lce$
- $e ightarrow le$
- $e ightarrow lce$
- $e ightarrow lce$
- $e ightarrow lce$
- $e ightarrow lce$
- $e ightarrow lce$

Preliminary Step:

Step 0':

- $S_a \rightarrow e$
- $e \rightarrow l$
- $e \rightarrow o$

Eliminate $A \rightarrow e$

Eliminate $A \rightarrow b$

Step 1:

- $S_c \rightarrow le$
- $S_c \rightarrow lce$

Add terminal symbol rules

Eliminate $A \rightarrow bc$

Step 2:

- $V_a \rightarrow c$
- $V_b \rightarrow l$

$ce \rightarrow V_b e o$
$ee \rightarrow V_a e o$
$ee \rightarrow V_b e e$
$ee \rightarrow V_a o o$
$ee \rightarrow V_a e o$
$ee \rightarrow V_b e e$
$ee \rightarrow V_a e o$
$ee \rightarrow V_b e e$
$S_a \rightarrow V_b e o$
$S_c \rightarrow V_a e o$

Step 3:

No rules to eliminate
8. Use the general stack-based algorithms to show:

- how the compiler will transform a Java expression \(8 - 4 - 2\) into inverse Polish notation, and
- how it will compute the value of this expression.

\[
\begin{align*}
8 & - 4 \rightarrow 8 4 - 2 \\
8 4 & - 2 -
\end{align*}
\]
9-10. Use a general algorithm for transforming PDA into CFG to design a CFG that corresponds to the following pushdown automaton. This automaton has two states: the starting state $s_a$ and the final state $s_b$. The transitions are:

- From $s_a$ to $s_a$, the transition is $a, \varepsilon \rightarrow 1$
- From $s_a$ to $s_b$, the transition is $b, 1 \rightarrow \varepsilon$.
- From $s_b$ to $s_b$, the transition is $b, 1 \rightarrow \varepsilon$.

Show, step by step, how the word $ab$ will be generated by the resulting grammar.
Tracing

S
\downarrow
A \rightarrow b
\downarrow
A \rightarrow a \rightarrow b
\downarrow
e
11 (for extra credit). For the context-free grammar from the Problem 3, show how the word 1010 can be represented as uvxyz in accordance with the pumping lemma.

\[ u = \epsilon \]
\[ v = 1010 \]
\[ w = \epsilon \]
\[ z = \epsilon \]
\[ \delta = \epsilon \]