You have one hour and 15 minutes to complete the exam. Do not open the exam until instructed to do so. Notes and texts are permitted. Computers, calculators, and communication devices are *not permitted*. Write all answers on the examination paper itself. BUDGET YOUR TIME WELL! *SHOW ALL WORK!*

<table>
<thead>
<tr>
<th>Question</th>
<th>[10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>[10]</td>
</tr>
<tr>
<td>Question 2</td>
<td>[10]</td>
</tr>
<tr>
<td>Question 3</td>
<td>[10]</td>
</tr>
<tr>
<td>Question 4</td>
<td>[10]</td>
</tr>
<tr>
<td>Question 5</td>
<td>[10]</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>[50]</strong></td>
</tr>
</tbody>
</table>

Bonus question [1 mark]: An example of a language $L \subseteq \{a\}^*$ that is not context-free is

$$\{a^n : n \text{ is a power of 2} \}$$

Question 1. [10 marks total]

(a) [4 marks] Give a context-free grammar in Chomsky normal form for \( \{w^Rw : w \in \{a,b\}^* \} \). (\(w^R\) is the reverse of \(w\).)

\[
S \rightarrow AX \mid BY \mid AA \mid BB \mid \lambda
\]
\[
T \rightarrow AX \mid BY \mid AA \mid BB
\]
\[
X \rightarrow TA
\]
\[
Y \rightarrow TB
\]
\[
A \rightarrow a
\]
\[
B \rightarrow b
\]

*The start variable cannot be recursive!*
(b) [6 marks] Let \( L = \{ w \in \{a,b,c\}^* : n_a(w) \geq n_b(w) + n_c(w) \} \). \( n_x(w) \) is the number of occurrences of symbol \( x \) in string \( w \). Give a context-free grammar for \( L \) in which the start variable is not recursive, and the only \( \lambda \)-production involves the start variable.

\[
S \rightarrow T / \lambda \\
T \rightarrow A / AB / BA / TT / ATB / BTA \\
A \rightarrow aA / a \\
B \rightarrow b / c
\]

The idea is to make sure that every \( b \) or \( c \) has a matching \( a \), that more \( a \)'s can appear anywhere, and that symbols can come in any order.

Question 2. [10 marks]
Let \( L = \{ w \in \{a,b,c\}^* : n_c(w) \geq n_a(w) \geq n_b(w) \} \). Is \( L \) context-free? Answer yes or no, and give a clear justification for your answer.

We use pumping to show that \( L \) is NOT context-free. So assume (to the contrary) that it is, and let \( p \) be its pumping constant.

Let \( w = a^pb^pc^p \) – note that \( w \) is in \( L \) and \( |w| \) is at least \( p \). Now write \( w = uvxyz \) with \( |vxy| \leq p \). Because \( vxy \) has length at most \( p \), if it contains an \( a \) it cannot contain a \( c \), and if it contains a \( c \) it cannot contain an \( a \).

Case 1: \( vy \) contains an \( a \) or a \( b \) but no \( c \). Then \( uv^2xy^2z \) either has more \( a \)'s than \( c \)'s or more \( b \)'s than \( c \)'s, and hence is not in \( L \), a contradiction.

Case 2: \( vy \) contains a \( c \). Then \( uv^0xy^0z \) has more \( a \)'s than \( c \)'s, and hence is not in \( L \), a contradiction.

But \( vy \) is not empty, so we have treated all the cases. Hence string \( w \) cannot be pumped, and \( L \) is not CF.

If you pick a string that contains MORE \( c \)'s than \( a \)'s or \( b \)'s, it can always be pumped, so that won't get the result that you want.

This is a somewhat different argument than for \( \{a^pb^pc^p\} \) because here in one case you MUST pump down. So the statement that it is "similar" is not enough, and the statement that it is the "same" is incorrect.
Question 3 [10 marks in total] Let G be the grammar with productions

\[
S \rightarrow aABb \mid A \mid B \mid D  \\
A \rightarrow aSAb \mid F \mid E \mid \lambda  \\
B \rightarrow ab \mid BF \mid DFD  \\
C \rightarrow BB \mid \lambda  \\
D \rightarrow EE  \\
E \rightarrow D \mid ED \mid DC  \\
F \rightarrow aBAb \mid FE \mid EF \mid \lambda
\]

(a) [5 marks] Which variables of G are nullable? useless? recursive?

<table>
<thead>
<tr>
<th>Variable</th>
<th>Nullable? Yes/No</th>
<th>Useless? Yes/No</th>
<th>Recursive? Yes/No</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>A</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>B</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>C</td>
<td>yes</td>
<td>yes not reachable</td>
<td>yes</td>
</tr>
<tr>
<td>D</td>
<td>no</td>
<td>yes not productive</td>
<td>yes</td>
</tr>
<tr>
<td>E</td>
<td>no</td>
<td>yes not productive</td>
<td>yes</td>
</tr>
<tr>
<td>F</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

(b) [5 marks] Modify the grammar so that (1) the start variable is not recursive, (2) there are no \(\lambda\)-productions except possibly \(S \rightarrow \lambda\), and (3) there are no unit productions.

*First I kill useless variables:*

\[
S \rightarrow aABB \mid A \mid B  \\
A \rightarrow aSAb \mid F \mid \lambda  \\
B \rightarrow ab \mid BF  \\
F \rightarrow aBAb \mid \lambda
\]

*Now I make the start variable not recursive, and then kill \(\lambda\)-productions:*

\[
S \rightarrow T \mid \lambda  \\
T \rightarrow aABB \mid A \mid B \mid aBb  \\
A \rightarrow aTAb \mid F \mid aTb \mid aAb \mid ab  \\
B \rightarrow ab \mid BF  \\
F \rightarrow aBAb \mid aBb
\]

*Now I kill unit productions:*

\[
S \rightarrow aABB \mid aTAb \mid aBAb \mid aBb \mid aAb \mid ab \mid BF \mid \lambda  \\
T \rightarrow aABB \mid aTAb \mid aBAb \mid aBb \mid aTb \mid aAb \mid ab \mid BF  \\
A \rightarrow aTAb \mid aBAb \mid aBb \mid aTb \mid aAb \mid ab  \\
B \rightarrow ab \mid BF  \\
F \rightarrow aBAb \mid aBb
\]

*And I am done.*
Question 4 [10 marks] Let \( L = \{c^n a^n b^n : n > 0\} \). Let \( B \) be the complement of \( L \).
Give a PDA that accepts the language \( B \). (You can specify the transition function of the PDA as a transition graph or as a table, as you prefer.)

Every string that contains substring \( ac, ba, \) or \( bc \) is in \( B \). Call this set \( B_1 \).
Every string with a different number of \( a \)'s and \( c \)'s is in \( B \). Call this set \( B_2 \).
Every string with a different number of \( a \)'s and \( b \)'s is in \( B \). Call this set \( B_3 \).
Then \( B \) is the union of \( B_1, B_2, \) and \( B_3 \).

All PDAs have start stack \( z \).
Here is a PDA for \( B_1 \): States \( \{p_0, p_1\} \), \( p_0 \) is start state, \( p_1 \) is (only) final state.
\( \delta(p_0, r, s) = \{(p_0, t)\} \) for \( (r, s, t) = (a, z, a), (b, z, b), (c, z, c), (a, a, a), (a, b, a), (b, b, b), (c, a, c), (c, b, c), \) and \( (c, c, c) \).
\( \delta(p_0, r, s) = \{(p_1, t)\} \) for \( (r, s, t) = (a, c, a), (b, a, b), \) and \( (b, c, b) \).
\( \delta(p_1, r, s) = \{(p_1, s)\} \) for \( r, s \) in \( \{a, b, c\} \).

Here is a PDA for \( B_2 \): States \( \{m_0, m_1\} \), \( m_0 \) is start state, \( m_1 \) is (only) final state.
\( \delta(m_0, r, s) = \{(m_0, t)\} \) for \( (r, s, t) = (a, z, az), (a, a, aa), (a, c, \lambda), (b, a, a), (b, z, z), (b, c, c), (c, z, cz), (c, a, \lambda), (c, c, cc) \)
\( \delta(m_0, \lambda, r) = \{(m_1, r)\} \) for \( r = a, c \).

Here is a PDA for \( B_3 \): States \( \{n_0, n_1\} \), \( n_0 \) is start state, \( n_1 \) is (only) final state.
\( \delta(n_0, r, s) = \{(n_0, t)\} \) for \( (r, s, t) = (a, z, az), (a, a, aa), (a, b, \lambda), (c, a, a), (c, z, z), (c, b, b), (b, z, bz), (b, a, \lambda), (b, b, bb) \)
\( \delta(n_0, \lambda, r) = \{(n_1, r)\} \) for \( r = a, b \).

And here is the PDA for \( B \): States \( \{q_0, n_0, n_1, p_0, p_1, m_0, m_1\} \), \( q_0 \) is start state, and \( \{n_1, p_1, m_1\} \) are final states. Take the union of the transitions above along with
\( \delta(q_0, \lambda, z) = \{(n_0, z), (m_0, z), (p_0, z)\} \).

Most of the credit is for figuring out that \( B \) is a simple union of CF languages.
The details of getting the PDAs exactly right are less crucial.
Question 5 [10 marks in total] In each part, answer yes or no, and explain. The grade is entirely for the explanation given, not the yes/no answer!

(a) [2 marks] Is there a CFL that is not a DCFL?

DCFLs are closed under complement, but CFLs are not (e.g., see Question 4). So there must be a CFL that is not a DCFL. Yes!

(b) [2 marks] Is there a DCFL that is not regular?

\(\{a^n b^n\}\) has a DPDA but by pumping it is not regular. So it is a DCFL that is not regular. Yes!

(c) [2 marks] Is there a linear language that is not a CFL?

A language is linear because it has a linear grammar. Every linear grammar is a context-free grammar as well. So every linear language is a CFL. No!

(d) [2 marks] Is there a CFL that is not linear?

The language of balanced parentheses is a CFL because it has an easy CFG \(S \rightarrow SS \mid (S) \mid \lambda\). But by the pumping lemma for linear languages, it cannot be pumped so it is NOT linear. Yes!

(e) [2 marks] Is there a finite language that is not linear?

Every finite language is regular (it’s easy to make a DFA for it). Every regular language has a right-linear grammar. Every right-linear grammar is a linear grammar. So every finite language is linear. No!

The fact that a language has a grammar that is not linear does NOT mean that the language is not linear! Maybe some other grammar for it is linear.