You have one hour and 15 minutes to complete the exam.
Do not open the exam until instructed to do so.
No notes, texts, computers, calculators, or communication devices are permitted.
Write all answers on the examination paper itself.
BUDGET YOUR TIME WELL!
SHOW ALL WORK!

<table>
<thead>
<tr>
<th>Question 1</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 2</td>
<td>15</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>Total</td>
<td>60</td>
</tr>
</tbody>
</table>

Sorted Grades out of 60 are 57 56 53 48 46 44 44 43 42 41 38 37 34 31 30 30 30 29 27 27 27 27 26 26 26 25 25 25 25 24 24 24 22 22 21 21 20 20 20 18 17 17 16 16 16 15 15 14 13 12 11 11 10 9 8. This midterm is worth 15% of your final grade. These grades will be scaled upwards as follows to determine what you get out of 15. If you received g/60, then compute 5 + (g/5) to get the grade out of 15. The new average is then 10.36/15 = 69.1%. The grade distribution out of 15 becomes 6.6 6.8 7 7.2 7.2 7.4 7.6 7.8 8 8 8.2 8.2 8.4 8.4 8.6 9 9 9.2 9.2 9.4 9.4 9.4 9.8 9.8 9.8 10 10 10 10.2 10.2 10.4 10.4 10.4 10.4 10.8 11 11 11 11 11.2 11.8 12.4 12.6 13.2 13.4 13.6 13.8 13.8 14.2 14.6 15.6 16.2 16.4
Question 1. [15 marks total]

Let G be the grammar

\[
S \rightarrow \varepsilon \mid aSb \mid SS \mid aAb \\
A \rightarrow \varepsilon \mid aAb \mid S
\]

(a) [3 marks] Give two leftmost derivations to show that G is ambiguous.

\[
S \Rightarrow SS \Rightarrow S \Rightarrow \varepsilon \\
S \Rightarrow \varepsilon
\]

These are two leftmost derivations of the same string in the language. Note that we must start with the start variable, and end with no variables (just terminals).

(a) [7 marks] Give an unambiguous grammar that is equivalent to G.

\[
S \rightarrow \varepsilon \mid E \\
E \rightarrow T \mid TE \\
T \rightarrow ab \mid aEb
\]

Alternatively here is an even more compact solution that one student found: \( S \rightarrow \varepsilon \mid aSbS \)

If we think of a as `(` and b as `)`, the language is the set of all strings with balanced parentheses.

(b) [5 marks] Give an equivalent grammar that is in Chomsky normal form.

Start with the grammar from (a). First add a new start variable R:

\[
R \rightarrow S \\
S \rightarrow \varepsilon \mid aSb \mid SS \mid aAb \\
A \rightarrow \varepsilon \mid aAb \mid S
\]

Now we kill \( \varepsilon \) rules. First we consider the rule \( A \rightarrow \varepsilon \). Look for A on the RHS of a rule and replace it in all possible ways. We form \( S \rightarrow ab \), \( A \rightarrow ab \) in this way. Then delete \( A \rightarrow \varepsilon \). Now consider \( S \rightarrow \varepsilon \). Replace it in all possible way to form \( S \rightarrow ab, S \rightarrow S, S \rightarrow \varepsilon, A \rightarrow \varepsilon \), and \( R \rightarrow \varepsilon \). Rule \( S \rightarrow ab \) is not added because it is already there. Rule \( S \rightarrow S \) is not added because the LHS and RHS are the same. Rules \( S \rightarrow \varepsilon, A \rightarrow \varepsilon \) are not added because these rules have been or are being deleted. So now we have

\[
R \rightarrow \varepsilon \mid S \\
S \rightarrow ab \mid aSb \mid SS \mid aAb \\
A \rightarrow ab \mid aAb \mid S
\]

The \( \varepsilon \) rules are ok, and there is no R on the right hand side of a rule. To kill the unit rule \( A \rightarrow S \),

\[
R \rightarrow \varepsilon \mid S \\
S \rightarrow ab \mid aSb \mid SS \mid aAb \\
A \rightarrow ab \mid aAb \mid aSb \mid SS
\]

To kill the unit rule \( R \rightarrow S \),
R → ε | ab | aSb | SS | aAb
S → ab | aSb | SS | aAb
A → ab | aAb | aSb | SS

Now get only variables or terminals on right hand sides
R → ε | XY | XSY | SS | XAY
S → XY | XSY | SS | XAY
A → XY | XSY | SS | XAY
X → a
Y → b

Last break up rules with three or more variables on the right hand side.
R → ε | XY | XP | SS | XQ
S → XY | XP | SS | XQ
A → XY | XP | SS | XQ
P → SY
Q → AY
X → a
Y → b

Done.

This could be simplified a LOT by noticing that the A variable is not needed at all!
R → ε | XY | XP | SS
S → XY | XP | SS
P → SY
X → a
Y → b
Question 2. [15 marks total] Any nondeterministic FA (with \( \varepsilon \) transitions) accepts a language that is also accepted by some DFA.

(a) [10 marks] Given a nondeterministic FA as input, sketch an algorithm that produces a DFA that accepts the same language.

Let \((Q, \Sigma, \delta, s, F)\) be the NFA.

For every state \(q\) in \(Q\), compute \(\text{Closure}(q)\) as follows. Initially it is \(\{q\}\). Then whenever \(r\) is in \(\delta(t, \varepsilon)\) and \(t\) is in \(\text{Closure}(q)\), add \(r\) to \(\text{Closure}(q)\). Repeat until no new states are added.

Now we build the DFA \((P(Q), \Sigma, \delta', s', F')\) where \(P(Q)\) is the powerset of \(Q\).

We set \(s' = \text{Closure}(s)\), and we set \(F' = \{S \subseteq Q : S \cap F \text{ is not empty}\}\).

Now for every \(S \subseteq Q\) and every \(a\) in \(\Sigma\) we set \(\delta'(S, a)\) to be the union of \(\text{Closure}(r)\) for every state \(r\) for which \(r\) is in \(\delta(s, a)\) for some \(s\) in \(S\).

By construction, this is a DFA. We can simplify by not generating a state \(S \subseteq Q\) if there is no way to get to \(S\) by any sequence of transitions from \(s'\).

[5 marks] Apply your algorithm to the machine \((Q = \{1, 2, 3, 4, 5\}, \Sigma = \{\text{a,b}\}, \delta, 1, \{2\})\), where \(\delta\) is given by the following transitions:

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>New State</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>(\varepsilon)</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>(\varepsilon)</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>(\varepsilon)</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>(\varepsilon)</td>
<td>3</td>
</tr>
</tbody>
</table>

You do not need to minimize the resulting DFA, but can do so if it is easier.

To simplify, delete state 5 because it is not an accept state and there is no way to leave it. Then compute \(\text{Closure}(1) = \{1, 2, 3, 4\}\), \(\text{Closure}(2) = \{2, 3, 4\}\), \(\text{Closure}(3) = \{2, 3, 4\}\), and \(\text{Closure}(4) = \{2, 3, 4\}\).

The start state is \(\{1, 2, 3, 4\}\).

1. \(\delta'(\{1, 2, 3, 4\}, \text{a}) = \{2, 3, 4\}\) and \(\delta'(\{1, 2, 3, 4\}, \text{b}) = \{2, 3, 4\}\).
2. \(\delta'(\{2, 3, 4\}, \text{a}) = \{2, 3, 4\}\) and \(\delta'(\{2, 3, 4\}, \text{b}) = \{2, 3, 4\}\).

This is a DFA with two states in which both are accepting. In fact the language accepted is ALL POSSIBLE strings over \{a,b\}.  

Question 3 [10 marks] Build a DFA to accept

(a) [5 marks] the language over \{a,b,c\} of strings that contain no two consecutive consonants.

<table>
<thead>
<tr>
<th>State r</th>
<th>(\delta(r,a))</th>
<th>(\delta(r,b))</th>
<th>(\delta(r,c))</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>s</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>t</td>
<td>s</td>
<td>z</td>
<td>z</td>
</tr>
<tr>
<td>z</td>
<td>z</td>
<td>z</td>
<td>z</td>
</tr>
</tbody>
</table>

Accept states: \{s,t\}. Start state: s.

(b) [5 marks] the language of strings over \{a,b\} in which the first and last symbols are different.

Assumption: This means that there is at least one symbol.

<table>
<thead>
<tr>
<th>State r</th>
<th>(\delta(r,a))</th>
<th>(\delta(r,b))</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>t</td>
<td>x</td>
</tr>
<tr>
<td>t</td>
<td>t</td>
<td>u</td>
</tr>
<tr>
<td>u</td>
<td>t</td>
<td>u</td>
</tr>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
</tr>
<tr>
<td>y</td>
<td>y</td>
<td>x</td>
</tr>
</tbody>
</table>

Accept states: \{u,y\}. Start state s.

Question 4 [10 marks] Give a regular expression for

(a) [4 marks] the set of all strings over \{a,b\} containing an even number of a’s.

\(b^*(ab^*ab^*)^*\)

(b) [6 marks] the set of all strings over \{a,b,c\} in which the total number of a’s plus the total number of c’s is odd.

\(b^*(a^uc^c) (b^*(a^uc^c) b^*(a^uc^c))^* b^*\)
Question 5 [10 marks] A context-free grammar \((V, \Sigma, R, S)\) is *regular* when every rule in \(R\) has one of the forms

- \(A \rightarrow a\) with \(A \in V\) and \(a \in \Sigma\);
- \(A \rightarrow aB\) with \(A, B \in V\) and \(a \in \Sigma\); or
- \(A \rightarrow \varepsilon\) with \(A \in V\).

Show that a language is regular *only if* it is generated by a regular grammar.

This is equivalent to the statement:

If \(L\) is a regular language, then \(L\) is generated by a regular grammar.

Suppose that \(L\) is a regular language. Then \(L\) is generated by a DFA \((Q, \Sigma, \delta, S, F)\). We form a regular grammar in which the variables are the states \(Q\), and the set of terminals is \(\Sigma\). Place the following rules in the grammar:

1. Whenever \(\delta(R, a) = T\), add the rule \(R \rightarrow aT\).
2. Whenever \(R\) is in \(F\), add the rule \(R \rightarrow \varepsilon\).

The start variable is \(S\).

Any derivation in the grammar corresponds to a computation of the DFA, and if the string derived has only terminals, the DFA must end in an accept state. So the language accepted by the DFA and the language generated by the grammar are the same.

Notes on Other Answers:

1. This is a question off homework 2, but there were very few correct answers.
2. Some students instead set out to show:

   If \(L\) is generated by a regular grammar then \(L\) is a regular language.

   This is not equivalent, but I gave almost full credit for a correct argument for this.
3. Some students instead set out to show:

   If \(L\) is not generated by a regular grammar, then \(L\) is not a regular language.

   As stated this is equivalent to what we want. However, most people who went this way changed the statement to the following, which is not equivalent:

   If \(L\) is generated by a grammar that is not regular, then \(L\) is not a regular language.

   It is not equivalent because the same language may be generated by two different grammars, one regular and one not. For example, the grammar with rules \(\{S \rightarrow \varepsilon, S \rightarrow aS, S \rightarrow bS\}\) is regular and generates the (regular) language \(\{a,b\}^*\). Nevertheless the grammar \(\{S \rightarrow \varepsilon, S \rightarrow aS, S \rightarrow bS, S \rightarrow SaabbabS\}\) is not regular even though it generates the same regular language. So the statement that “If \(L\) is generated by a grammar that is not regular, then \(L\) is not a regular language” is *false*. 