Model solutions (and some comments): Section A

1. a. The strings \{a, aa, aaa\} for example; alternatives are \(e\) or \(baa\) etc.

   b. Strings (over \(a\) and \(b\)) consisting of \(a\)'s (possibly empty), plus any other string with
      the feature that any \(b\) is followed by at least 2 \(a\)'s.

   c. An equivalent FA is obtained by just removing the \(a\) transition from \(q_1\) to \(q_2\). A regular
      expression could be \(\{a, baa\}^*\) or \(a^*(a^*baaa)^*\) or \(\{a^*, a^*aaa, a^*baa\}^*\)
      Not quite correct: \(a^*baaa^*\) or \(a^*[a, b]aaa^*\) or \(a^*(baaa)^*\) or \(a^*(baa)^*a^*\) or
      \(\{a, a, a, baa\}^*\) or \(a^*\{baa\}a^*\) or \(a^*\cup\{baaa\}^*\).

   d. This is basically bookwork but it was done quite poorly.

   Sets of states in the NFA correspond to individual states in the corresponding DFA;
   new state transitioned to from state \(q\) in the DFA is the union over all states of the sets
   of states labelled by that symbol in the NFA coming out of states in \(q\). Accept if any
   state in the union accepts.

2. a. Given a regular language, there exists \(N\) such that any word \(w\) in the language of
   length \(> N\) has a prefix \(w'\) of length \(N\), where \(w'\) has a non-empty substring that can
   be "pumped", ie by replacing that substring with any sequence of copies of it, one
   obtains other words in the language. A substring that can be pumped corresponds to
   a loop in the FA. (strictly, an accessible loop, but wouldn't require that to be pointed
   out.)

   b. For any \(N\), \(0^N\{0^N\}^*\) is in the language; assume \(N\) is large enough to apply the PL with
      that value of \(N\); then \(0^{N+r}0^{N-1}\) belongs to the language, where \(0^r\) can be pumped,
      and we have \(r\) is positive. But this can be seen to fail to comply with the description
      of strings in the language.

3. The question was not done well in the exam. Answers were commonly omitted or definitions
   of the relevant classes of languages were given, as opposed examples. Some confusion
   between "context-free language" and "context-free grammar"? "Deterministic" means
   that the language has a deterministic PDA, it does not mean that you can write down a CFG
   where each variable has only one rule.

   a. e.g. palindromes over \(\{a, b, c\}\) where there is a single occurrence of \(c\) in a word (by
      necessity, in the centre).

   b. e.g. \(\{0^n1^n : n \geq 1\}\), Palindromes, etc. Full credit requires formal definition
      of language (but not proof).

   c. e.g. \(\{a^nb^nc^n : n \geq 1\}\); \(\{1^p : p \text{ is a prime number}\}\), etc.

   d. Expect to see Halting Problem for Turing Machines,

      \(\{\eta(M)w, \eta(M)\text{ is encoding of TM, } M, \text{ and } M \text{ halts on } w\}\)

      but other (correct) answers acceptable.

4. This question was done well in the exam.
a.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>S → ABCS</td>
<td>S → ABCS</td>
<td>S → ABCS</td>
<td>S → d</td>
</tr>
<tr>
<td>A</td>
<td>A → aA</td>
<td>A → ε</td>
<td>A → ε</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>B → bB</td>
<td></td>
<td>B → ε</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td>C → c</td>
<td></td>
</tr>
</tbody>
</table>

b. FIRST(S) is all four letters. A and B are nullable.

c. S → ABCS → BCS → bBCS → bbCS → bbcS → bbcABCBS → bbcABCBS → bbcABCBS → bbcaS → bbcaSd
Section B

Answer two questions in this section.

1. A common error was to claim that states 4 and 8 are inaccessible, on the grounds that because you can’t accept after they have been reached, it is possible to remove them. They can be removed (given the convention that undefined transitions lead to rejection) but they are accessible.

   a. There are no inaccessible states; it is straightforward to check that we can reach any state starting at the initial one.

      Indistinguishable states: if we do this by working out what words are accepted starting from each state, we note that states 4 and 8 are all-rejecting, so they are indistinguishable.

      state 5 accepts the empty word and nothing else.

      state 7 accepts the empty word and nothing else, so same as state 5

      state 6 accepts strings of a’s

      state 2 accepts a ∪ ba∗.

      state 3 accepts ba∗ ∪ a, same as 2.

      state 1 accepts {a, b}L2 where L2 is the above language accepted by state 2.

      Generally, delete the inaccessible states and merge the indistinguishable states (combining transitions).

      A minimal DFA has state 1 with an a, b-transition to state 2; (state 3 is merged with state 2); state 2 has a b transition to state 6 and an a transition to state 5 (merged with 7); state 5 has a, b-transition to state 4 (merged with 8). States 4 and 5 are accepting.

   b. Suppose M has n states that are accessible and distinguishable.

      Suppose M’ is equivalent to M but has only n – 1 states.

      We can find n words that reach each of the n states of M

      use accessibility here

      Two of these words must reach the same state of M’.

      We can find a suffix for these two words such that M would accept one but not the other

      here we use distinguishability

      But both words, with any suffix attached, must reach the same state in M’. So M’ cannot accept the same set of words.

2. In part (a) of this question, a non-deterministic PDA was commonly given as an answer. The inattention to the requirement of being deterministic also affected part (b), where a common answer was “yes”, and a NPDA was given as solution. Part b(ii) was done well in general.

   a. States A, A′, I, F; stack alphabet x, y; I is initial state; F is accepting state, assume the criterion for accepting is that you must end in state F with an empty stack.

      δ(I, ε, ε) = (A, y) (mark the bottom of the stack)
\[ \delta(A, a, \epsilon) = (A, x) \text{ (add stack symbol for each } a \text{ preceding } b) \]
\[ \delta(A, b, \epsilon) = (A', \epsilon) \]
\[ \delta(A', a, x) = (A', \epsilon) \text{ (remove stack symbol for each } a \text{ following } b) \]
\[ \delta(A', a, y) = (A', y) \text{ (allow excess stack contents to be discarded)} \]
\[ \delta(A', b, y) = (F, \epsilon) \]

A, perhaps nicer, approach is to allow the final state to pop stack symbols, so it can empty the stack.

b. (i.) Not accepted by a deterministic PDA. Consider words such as \(a^m ba^n ca^m ba^n\). On reaching the central \(c\), a DPDA has to have an empty stack since it should be able to accept at that point. Consequently it cannot remember the value \(n\), and the entire word should be accepted provided that \(m = n\), so in fact it is necessary to remember the number \(n\).

(ii.) \(S \rightarrow aSA; S \rightarrow bSB; S \rightarrow cSC;\)
\(A \rightarrow a; B \rightarrow b; C \rightarrow c;\)
\(S \rightarrow a[b|c|e]\)

3. Not many attempts at giving an unrestricted grammar: CFGs were offered, which usually generated \(a^*b^+c^+\). TMs were offered a couple of times, which is arguably harder than writing down an unrestricted grammar. Problem here was probably forgetting what is meant by “unrestricted grammar”.

a. \(S \rightarrow XABCZ\)
\(X \rightarrow F\)
\(FA \rightarrow F; FB \rightarrow F; FC \rightarrow F;\)
\(Fa \rightarrow aF; Fb \rightarrow bF; Fc \rightarrow cF; FZ \rightarrow \epsilon\)
\(X \rightarrow R\)
\(RA \rightarrow AaR; Ra \rightarrow aR, \text{ and similarly for } B, C\)
\(RZ \rightarrow LZ\)
\(\alpha L \rightarrow L\alpha\) for all symbols \(\alpha\) other than \(A\)
\(AL \rightarrow AX\)

\(S \rightarrow XABCZ \rightarrow RABCZ \rightarrow AaRBCZ \rightarrow AaBbRCZ \rightarrow AaBbCcRZ\)
\(\rightarrow AaBbCcLZ \cdots \rightarrow ALaBbCcZ \rightarrow XAaBbCcZ\)
\(\rightarrow RAaBbCcZ \rightarrow AaRaBbCcZ \rightarrow AaaRBbCcZ \rightarrow AaaBbRbCcZ\)
\(\rightarrow AaaBbRbRCZ \rightarrow AaaBbbCcRcZ \rightarrow AaaBbbCcRcRZ \rightarrow AaaBbbCcRcLZ\)
\(\ldots \rightarrow ALaaaBbbCcZ \rightarrow XAaaaBbbCcZ \rightarrow FAAaaBbbCcCZ\)

(the above produces the \(F\) that initiates the final deletion of variables)

b. bookwork.