I. Submit your work to the course gradescope server following the course submission requirements.
If this assignment does not have a coding requirement, simply provide your typeset work in an archive file within a `./doc` directory.

II. Typesetting should meet the common expectation for professional and academic work.

III. "Pseudo code" is a vague term; imagine writing the pseudo code for another programmer who is familiar with the conventions of imperative languages, **but doesn't know any of the same languages you know**. In other words, avoid the use of `==`, `i++`, `x-=y`, `&&`, `range(N)`, `lst[:n:i]`. Pseudo code can **always use** mathematical notation: $\min\{\cdots\}, \lfloor\cdot\rfloor, \lceil\cdot\rceil, \text{mod}, \ldots$

IV. No part of your submitted work should be generated or informed by "AI" in any of its forms.

---

§4.1 of the text asserts that regular languages are a proper subset of the languages described by context free grammars. The text's claim being that any regular language can be defined by a context free grammar with rules **of the form**:

$$\begin{array}{rcl} \mathit{START} & \rightarrow & A\ \$ \\ & \vdots & \\ A & \rightarrow & \tau\ \beta \\ A & \rightarrow & \pi \\ & \vdots & \end{array}$$

Where the grammar symbols A, β are singular non-terminals ($\in N$) and τ, π are singular elements $\in \Sigma_\lambda$.

1. (30 points) Given a regular expression, NFA or DFA describing a regular language over the alphabet Σ, show a construction algorithm that produces a grammar $G(\mathit{START}, \Sigma, N, P)$ where N is the set of non-terminals and P is the set of production rules.

   a. You may assume the NFA and DFA representations of the language do not have dead or unreachable states.

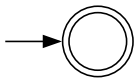
   b. You may cite any algorithm presented in lecture without evidence of its correctness.

   c. Your procedure should accept as parameters the regular language definition (in a representation of your choice), its Σ and the starting goal to use for the grammar. Your procedure should return the constituent parts of the constructed grammar, namely: $\mathit{START}, \Sigma, N$, and P

Meets requirements: +10, Correctness: +10, Clarity: +10

2. (10 points) Show the grammar G your algorithm produces for the following regular languages:

   (a) $\Sigma = \{q, r, s, t\}$



   (b)

	State	a	b	c	d	e	f
	0		8	1	2		
	1			4	2	5	
	2		9				9
	3		9				
	4			4	2		
	5	8	10				9
	6	9			3		
	7					5	
+	8	0				7	
+	9				9	3	
+	10	0		6		7	

   (c) $\lambda \mid x^+(g^+x^+)* \mid g^+(x^+g^+)*$

2 out of 3: +8, 1 out of 3: +5

3. (30 points) Finally, argue that **any** grammar G your algorithm constructs does in fact describe an **unambiguous**[1] **LL(1) language**.

Meets requirements: +10, Correctness: +10, Clarity: +10

[1] Absent of common grammer faux pauxs such as $A \rightarrow Q\ x\ Q$ that might permit two different parse trees for one input.