CSE396, Spring 2026 Problem Set 1 Due Fri. 2/6, 11:59pm

Reading: Tuesday's lecture will begin covering *NFAs* and *regular expressions* in tandem. The text delays the formal definition of regular expressions until section 1.3. Please first read section 1.2 plus the part of section 1.3 that defines regular expressions. Then note that the text divides the treatment of closure under regular operations between the same sections. Please then read that treatment in one piece—*skipping over* the theorem about converting an NFA into an equivalent DFA for now. Stop before reading the topic of "generalized NFAs (GNFAs)" in section 1.3 (p69 on).

Homework—part online (TopHat), part written, and all *individual work*:

(1) Using *TopHat*, the worksheet titled **S26 HW1 Online Part**. There are 10 questions, each worth 2 points, for 20 total. *(The 11:59pm deadline on TopHat has no "stretch.")*

The other two problems are to be submitted as PDFs using the *CSE Autograder* system. Scans/photos of handwritten sheets are fine provided they are *easily legible* and *do not have excessive file-size.* Or you may type your answers.

(2) This question reviews the concept of an equivalence relation from CSE191 or other discrete mathematics course, as it will soon resurface in the treatment of nonregular languages, and other concepts that may come in handy later. Let $G = (V, E)$ be the graph with vertices $V = \{2, 3, 4, 5, 6, 8, 9\}$ and edge set E consisting of all pairs (u, v) such that the numbers u and v share a factor other than 1. (That is, u and v are *not* relatively prime.) Draw a picture of G (6 pts.) and then answer the following short questions:

(a) Is the edge relation E reflexive? (2 pts.)

(b) Is E symmetric? (2 pts.)

(c) Is E transitive? If you say *no*, please give a counterexample involving three different vertices. If you say *yes*, say why you are convinced of that. (6 pts.)

(d) Is there a vertex w that does not have an edge to any *other* vertex? (2 pts., for 18 total)

(3) Text, exercise 1.5, parts (d) and (g) only. That is:

(d) Design a DFA M that accepts all and only those binary strings that *don't* consist of zero-or-more 0s followed by zero-or-more 1s.

(g) Design a DFA M that accepts all and only the strings that *don't* have exactly two 0s in them.

In both cases the alphabet is $\Sigma = \{0, 1\}$. The text suggests that you first design a DFA M' for the complements of these languages—that is, changing "*don't*" to "*do*" in the stated conditions—then make M by swapping accepting and rejecting states. But you can also treat this as a case of rewriting the definitions themselves to be positively stated instead—for (d), think of what's needed to make the condition fail, and for (g), think of categories "zero," "one," "two," and "three-or-more." However you do it, full credit requires explaining the logic via a well-commented arc-node diagram. (12+12 = 24 pts., for 62 total on the problem set)