CSE396, Spring 2026 Problem Set 7 Due Mon. 4/6, 11:59pm

This homework is due on **Monday**, April 6, not on Friday. Prelim II has definitely been set for **Tuesday, April 21**, in class period. Problem 3(b) will be covered in the Tuesday 3/31 lecture—see notes at https://cse.buffalo.edu/ regan/cse396/CSE396S26Week10Tue.pdf.

Reading

We have turned the corner to the final third of the course, on general computability. This begins with technical details of various kinds of Turing machines. The emphasis will however be *moving away from* low-level details. The detail of Turing machines works to establish that they can:

- copy and compare strings;
- find a matching (sub-)string on a tape;
- do basic arithmetic using binary strings (as already shown in the "$3n + 1$" example);
- branch according to what char is read at a given time;
- execute loops, especially data-dependent (i.e., un-counted) loops.

These operations suffice for universal computation—indeed, to build a reasonably efficient assembly simulator. The text touches on this at the end of Chapter 3; my handwritten Universal RAM Simulator sketch makes it more concrete. *Once we have done this, the character-level details matter less.* We can mostly talk interchangeably about "Turing Machines" and the high-level programming languages we know and love, and discuss computational problems beginning in Chapter 4 from the latter's standpoint.

Also note that the text chapters themselves become shorter. Thus, please read Chapter 3 and look ahead to the first section of chapter 4. Next week will focus on *undecidable* problems from the rest of chapter 4. The other thing to bear in mind is that the one technical detail of Turing machines that we will care about is whether they have one tape or two-or-more tapes, along with possible restrictions on what they can do on their tapes. This begins attention to the *computational complexity* of problems, not just polynomial versus exponential time, but also linear versus quadratic-or-more time.

———-**Homework**—part online and all *individual work*—due **Mon. 4/6, 11:59pm**———-

(1) Using *TopHat*, the "Worksheet" titled **Spr 2026 HW7 Online Part**. There are 10 questions, each worth 2 points, for 20 total. (Recall the TopHat deadline is strict.)

(2) Over $\Sigma = \{a, b\}$, define $L = \{ww^R : \#a(w) = \#b(w)\}$, where w^R means the string w written in reverse, e.g., $aaba^R = abaa$. Prove using the CFL Pumping Lemma that L is not a CFL. *Hint:* Think of test strings $x \in L$ that have the form $x = a^i b^j a^k$. What must then be true about the numbers i, j, and k? Please show the full script of the CFL PL proof, though you may abbreviate case(s) that are similar to a previous one. (24 pts.)

(3) Now let $\Sigma = \{a, b, \#\}$ where '#' is a special divider symbol. Consider the language

$$A = \{u\#v : u, v \in \{a, b\}^* : |u| = |v| \ \wedge \ \#a(u) = \#a(v)\}.$$

Put another way, a string x belongs to A if and only if it has odd length with a single # sign that is in the middle, and the number of a's before the # equals the number of a's after it.

(a) Prove that A is not a CFL. It may help to find a regular language R such that $A \cap R$ is simpler to handle with the CFL Pumping Lemma. Proving that $A \cap R$ is not a CFL then carries the same conclusion for A. (24 pts.)

(b) Design a *two-tape* Turing machine M such that $L(M) = A$. For some hints, have M make two "passes" over the input x on tape 1: one to verify that it contains a single # that is exactly in the middle, and a second pass (which can be right-to-left) that compares the number of a's before and after it. Explain in comments how the second tape is used to make both passes run in time linear in $n = |x|$, and say exactly where your code deviates from the restrictions that define a DPDA on both tape 1 and tape 2. (24 pts., for 92 total on the set)