Reading:

Finish chapter 3 and look back at just these parts of the otherwise-skipped sections 2.2 and 2.4: the description of PDAs and the definition of DPDA on the first two pages of section 2.4—enough to satisfy you that the official Turing-machine based definitions in this course are equivalent. Also read the following two new handouts of examples which will be covered in Tuesday’s lecture:

http://www.cse.buffalo.edu/ regan/cse396/TwoTapeTMs.pdf
http://www.cse.buffalo.edu/ regan/cse396/UTMRAMsimulator.pdf

OK, just skim the latter—the point is not the details of how it could be completed, but satisfaction that it can be done, so that Turing Machines can simulate assembly code and hence be a compilation target for any high-level language. This is IMHO the most concrete evidence for the Church-Turing Thesis which will be discussed in the lecture—and which we like the text will use only to justify the informal/pseudocode description of machines which will begin in Chapter 4. Then move into Chapter 4 proper for Thursday’s lecture. Stop short of the text’s discussion of Cantor’s Theorem—diagonalization and undecidability will come the following Tuesday lecture before Prelim II.

Chapter 3 is the last material in the domain of Prelim II—which is however “cumulative,” i.e., it includes material from Chapters 0 and 1. Again, it is being held on Thursday, April 27, in class period, with the same rules (one notes sheet etc.) as Prelim I.

Assignment 8, due in hardcopy and in class Thu. 4/20*

(**except R4 may submit Fri. 4/21 before 2pm to my Davis 326 office)

Please write your name, Student ID#, and recitation attended atop your HW.

(1) This problem aims to cement your understanding of (D)PDAs between the text and the Thursday 4/13 lecture (whose notes have been posted).

(a) The DPDA $M$ accepting $L = \{a^n b^n : n \geq 0\}$ at the bottom of page 3 at the end of the notes has a bug as drawn. The bug (see note added there also) is that if $x = a^n$ where $n \geq 1$, then $x \notin L$, but $M$ still accepts $x$ because the loop at the start state will still see a blank $B$ on tape 2 after reading the $a$’s. Re-design $M$ to fix this bug. The most thematic fix is to take the hint in my lecture to have the first step introduce a bottom-of-stack marker $\land$ on tape 2. That first step can stay stationary on the first char $x_1$ of the input $x$ on tape 1. (In the first example at http://www.cse.buffalo.edu/ regan/cse396/TwoTapeTMs.pdf, I chose to reinforce the “column 0” visual by having the input tape head start there too, that is on the blank to the left of $x_1$ and moving right at the first step. But it is fine to start it on $x_1$ instead; if $x = \epsilon$ it will be scanning a blank on tape 1 in either case. 9 pts.)

(b) Suppose $c \in \Sigma$, $A \in \Gamma$, and a state $p$ are such that a DPDA $M$ has an instruction $(p, (c, A), \ldots)$ but goes into an “infinite loop” upon executing it. That is, $M$ thenceforth never makes a $R$ move beyond the char $c$ on the input tape, and never moves its second tape head left of the cell occupied by the $A$. (It might push other chars after the $A$ and the TM version is also allowed to change $A$ in one step, but it never pops whatever is there.) Then you can
replace the instruction by \((p, (c_A), (c_B), (R), d)\) where the state \(d\) acts like a dead state: it cleans up tape 2 by continuing to pop until it’s blank, reads tape 1 until it hits the blank at the end, and finally goes to \(q_{rej}\) on \((c_B)\). The new machine \(M_0\) obtained by doing this replacement in all looping cases of \(p, c, A\) gives \(L(M_0) = L(M)\) without ever looping.

Go on to describe how to modify \(M_0\) into a DPDA \(M'\) such that \(L(M')\) equals the complement of \(L(M)\). For full credit, you should explain that if \(M\) obeys “good housekeeping” on all of its non-looping computations, then \(M'\) obeys it on all computations. “Good housekeeping”—as defined in my lecture notes and applying to all TMs not just PDAs—means that all halting computations end up by erasing tape(s) 2 (and beyond), reading all of tape 1 and ending with the tape head on the cell to the right of the end of the input \(x\) (which could hold a special end-marker \$ char instead of \(B\) if you wish—this is an option also in part (a) above), and either rejecting in the state \(q_{rej}\) or accepting in \(q_{acc}\). In particular, a “good housekeeping” PDA simultaneously obeys both the “accept by final state” and “accept by empty stack” criteria—well for the latter we technically need to make the final arc(s) to \(q_{rej}\) refuse to pop the bottom-of-stack marker \(\land\). Finally observe how this shows that the class of DCFLs—unlike the class of CFLs—is closed under complementation. (9 + 3 = 12 pts., for 21 on the problem)

(2) Define \(L = \{wc^mw : w \in \{a,b\}^*, m = |w|\}\). This is the same as the language on Assignment 7 (problem 3) except that the final \(w\) is not reversed. Design a two-tape TM \(M\) such that \(L(M) = L\). Note that since \(m = 0\) is allowed by default, \(\epsilon\) does belong to \(L\), so you will have to handle that case. Also indicate exactly where your \(M\) fails to be a PDA and answer whether you ever needed to keep the input head stationary in any step. (18 + 3 + 3 = 24 pts., for 45 total on the set)