University of Nevada, Las Vegas Computer Science 456/656 Spring 2024

Assignment 7: Due Saturday April 27, 2024, 11:59 PM

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet. You will receive a message from the graduage assistant, Zachary Edwards, telling you how to turn in the assignment.

Of course, you can answer some of these questions by simply copying from handouts or nodes, without any understanding. But I advise against that.

1. Prove that every decidable language is enumerated in canonical order by some machine.

2. Prove that every language that is enumerated in canonical order by some machine is decided by some other machine..

3. Prove that eny language accepted by any machine can be enumerated by some other machine.

4. Prove that any language which is enumerated by some machine is accepted by some other machine.

5. I have repeatedly stated in class that no language that has parentheses can be regular. For that to be true, there must be parenthetical strings of arbitrary nesting depth. (If you don't know what nesting depth is, look it up.)

   Some programming languages have limitations on nesting depth. For example, I have read that ABAP has maximum nesting depth of 256. (Who would ever want to go that far!)

   The Dyck language is generated by the following context-free grammar. (As usual, to make grading easier, I use a and b for left and right parentheses.)

   1. $S \rightarrow aSbS$
   2. $S \rightarrow \lambda$

   (a) Use the pumping lemma to prove that the Dyck language is not regular.

   (b) Let D be any finite integer. Let L be the language consisting of all members of the Dyck language whose nesting depth does not exceed D. Prove that L is regular.

6. We known that context-free languages are exactly those which are accepted by push-down automata. We now define a new class of machines, which we call "limited push-down automata." An LPDA is exactly the same as a PDA, but with the restriction that the stack is never allowed to be larger than some given constant. What is the class of languages accepted by limited push-down automata? Prove your answer.

7. Prove that every context-sensitive language is decidable. The way to do this is to start with an arbitrary context-sensitive grammar, using the definition I gave in class (that's not the only definition) namely that the right side of any production must be at least as long as the left side, and then design a program which decides whether any given string is generated by that grammar. (If the string has length n, the running time of your program could be very long, maybe an exponentially bounded function of n?)