81 pts: Finals Quiz (3 pts, 27 qns); 19 for Long Answer (1, 9, 9 pts)

Please choose up to two options per question. For each correct option chosen, you will earn 1.5 points. Please do not choose more than two options—you will then get zero points for the whole question. All questions as well as the two correct options in each question have the same point value. Thus, (1) Choosing two wrong options results in zero points for the question; (2) choosing one right and one wrong results in 1.5 points (for the right option).

1. Consider these two regular expressions:

    $R_1 = (01^*0)^*$
    $R_2 = (0^*1^*0)^*$

    Let $L(R)$ mean the language of regular expression R.

    Choose from this list:

    (a) $L(R_1)$ contains 1010 ☐

    (b) $L(R_2)$ contains 1010 ☐

    (c) $L(R_2)$ contains 1110 ☐

    (d) $L(R_1)$ contains 000 ☐

2. Consider the same regular expressions, i.e., $R_1 = (01^*0)^*$ and $R_2 = (0^*1^*0)^*$: Consider the DFAs presented in Fig-A and Fig-B (states that do not fully decode are assumed to transition to a black-hole state). Also let $L(A)$ mean "the language of Fig-A's DFA" and $L(B)$ mean "the language of Fig-B's DFA". Now choose your answers:

Fig-A: Fig-B:

    (a) $L(A) = L(R_1)$ ☐
    (b) $L(B) \subseteq L(R_1)$ ☐
    (c) $L(A) \cap L(B) \cap L(R_1) \cap L(R_2)$ is infinite ☐
    (d) $L(B) \subseteq L(A)$ ☐

3. Consider the PDA "X" below and let its language be $L(X)$. Let P stand for the polynomial-time language family (in the sense of complexity). Select your options:

a, a ; aa
b, a ; "
b, # ; b#
a, b ; "
a, # ; a#
b, b ; bb

```
   ──▶( I )  ", # ; "  ▶(( F ))
```

:
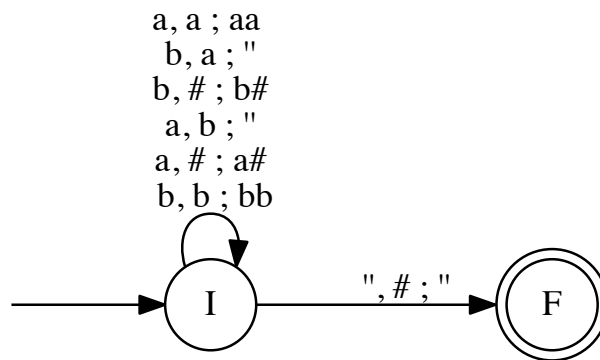
(a) $bab \in L(X)$ ☐

(b) X can be recommended as a PDA for recognizing $\{a^n b^n \; : \; n \geq 0\}$. ☐

(c) $baab \in L(X)$ ☐

(d) $L(X) \in P$ ☐

4. Consider the following CFG

```
S -> M C | A N
M -> a M b | ''
N -> b N c | ''
A -> a A | ''
C -> c C | ''
```

Choose answers from this list:

(a) L(S) is regular. ☐

(b) A direct CFG to PDA conversion of this grammar will yield a "working" state that has a self-loop labeled with more than 10 cases ☐

(c) This grammar is ambiguous but it can also be disambiguated ☐

(d) abbcc is in the language of this grammar ☐

5. Pick all true assertions (One convention: we always use the tightest language classification; thus if a language is not regular, then only will we begin calling it context-free, etc.)

   (a) Recursively Enumerable languages are closed under complementation   □

   (b) Recursively Enumerable languages are closed under intersection   □

   (c) Context-free languages are not closed under concatenation   □

   (d) Every recursively enumerable language is infinite   □

6. These are true of Lambda conversions.

   (a) $(\lambda x.(\lambda y.(y + x))) \rightarrow (\lambda x.(\lambda x.(x + x)))$ via the α rule.   □

   (b) $(YY) \rightarrow Y(YY)$ □

   (c) $(\lambda x.x)(\lambda y.y) \rightarrow (\lambda z.z)$ via α and β rules.   □

   (d) The combinators S, K, and I are a Turing-complete; we cannot model any one of these combinators in terms of the other two.   □

7. Given these languages, pick the correct answers from the list below:
   - $L_1 = \{w \in \{a, b\}^* : \#_a(w) < \#_b(w)\}$
   - $L_2 = \{(ab)^n : n \geq 0\}$
   - $L_3 = \{a^{n^2} : n \geq 0\}$
   - $L_4 = \{a^i b^j c^k : \text{if } (i = 3) \text{ then } (j = k)\}$

   (a) L_1 and L_4 are context-free.   □

   (b) L_2 and L_3 are context-free.   □

   (c) L_2 and L_4 are context-free.   □

   (d) One of the languages in this list is not Recursive.   □

8. Let the language (over $\Sigma = \{0, \#\}$) defined by the following CFG be called L.

```
S -> T T | U
T -> 0 T | T 0 | #
U -> 0 U 0 0 | #
```

These are true of L's language:

   (a) L is regular.   □

   (b) L is context-free.   □

   (c) L would be regular if `S` goes to `T` instead of to `TT`.   □

   (d) The language of `T` is not regular, as it has both types of linearity.   □

9. Identify the correct answers

(a) If a language L is NP-Complete and $\overline{L}$ is in NP, then for every language $L' \in NP$, it is also the case that $L' \in CoNP$ (and vice-versa). ☐

(b) The complexity of 3COLOR is exponential while that of 2COLOR is polynomial ☐

(c) For a new language X, if we can show that $3SAT \leq_P X$, then we still do not know whether X is NP-complete ☐

(d) All functions have exactly one fixpoint ☐

10. Choose the correct facts from this list:

(a) A Turing machine accepts an input only when it has read its input fully and the control state goes from its starting state q0 in Q to some terminal state f in F (from f, there are no further moves) ☐

(b) In general, it is not possible to tell whether a TM will ever read its input fully ☐

(c) A Turing machine started in state q0 in Q accepts its input when it attains the state f in F from which it has no further moves ☐

(d) A Turing machine started in state q0 in Q accepts its input if it attains the state f in F at least once ☐

11. Choose from the list below those languages that are RE but not recursive:

(a) The language $Halt_{TM}$ which is the set of machine, word pairs $\langle M, w \rangle$ such that Turing machine M halts on word w ☐

(b) The language { } ☐

(c) The set of Turing machine descriptions whose language contains the string 101 ☐

(d) The language of Turing machines whose tape has a fixed length. ☐

12. Choose from these facts:

(a) Every NP-complete language is recursively enumerable ☐

(b) If an NDTM has a language-equivalent DTM, then P=NP ☐

(c) Every regular language L is in P because there is a deterministic P-time decider for it ☐

(d) In the Chomsky hierarchy we studied consisting of DFA, PDA, and TM, all have non-deterministic versions that can be converted to corresponding deterministic versions ☐

13. Let us consider the Post correspondence problem (PCP). One can enumerate all PCP instances that have solutions. Given this, choose all the assertions below that are true.

The set of PCP instances that have solutions is:

(a) Not even recursively enumerable (a procedure does not exist) ☐

(b) Not recursive (no algorithm exists) ☐

(c) Recursively enumerable (a procedure exists) ☐

(d) Context-free (a CFG parser exists) ☐

14. In computability theory, when we are creating "a mapping reduction," the following are implied:

(a) We are showing that a "new problem" is being reduced to an old, and impossible-to-solve problem. ☐

(b) We are showing that given an old impossible-to-solve problem "A", a mapping can be created to the new problem "B" that is at least as hard as A. ☐

(c) Both the "old problem" and the "new problem" must be equally hard to solve. ☐

(d) If a decider is shown to exist for the new problem, that would result in a contradiction. ☐

15. Suppose someone produced a proof, from first principles, for A_{TM} being non-recursive. They want to use that to show that $Halt_{TM}$ is non-recursive via a mapping reduction function f. The following two steps accomplish a sound reduction from A_{TM} to $Halt_{TM}$.

(a) Show that for every $\langle M, w \rangle$ pair, $\langle M, w \rangle \in Halt_{TM}$ iff $f(\langle M, w \rangle) \in A_{TM}$ ☐

(b) Show that for every $\langle M, w \rangle$ pair, $\langle M, w \rangle \in A_{TM}$ if $f(\langle M, w \rangle) \in Halt_{TM}$ ☐

(c) Show that for every $\langle M, w \rangle$ pair, $\langle M, w \rangle \in A_{TM}$ iff $f(\langle M, w \rangle) \in Halt_{TM}$ ☐

(d) Show that function f is computable ☐

16. Consider the following claimed mapping reduction proof to show that checking whether a TM has a *context-free language* is not decidable. Let a decider for a context-free language, if one exists, be called D_{cfl}.

```
M'(x) {
   if x is of the form 0^n 1^n then goto accept_M' ;
   Run M on w ;
   If M accepts w, goto accept_M' ;
   If M rejects w, goto reject_M' ; }
```

Pick out the two correct arguments from below:

(a) Under the condition "M does not accept w," the language of M' is context-free; Under the condition "M accepts w," the language of M' is Σ^* which is regular. Thus, the existence of D_{cfl} allows us to tell whether the language of M' is a CFL or not, and hence decide A_{TM}. ☐

(b) Under the condition "M rejects w," the language of M' is also context-free. This makes it impossible to finish the "if and only if" of a mapping reduction. ☐

(c) If we make the initial test
if x is of the form ``ww'' then goto accept_M'
the proof would be fixed. Here, w is a member of $\{0, 1\}^*$. ☐

(d) We must make the initial test
if x is of the form ww^R then goto accept_M'
to fix the proof. ☐

17. Let *PCP* be all PCP instances that have a solution. Then:

(a) In PCP, if the tiles are required to be used *without* repetition, the problem will still be undecidable ☐

(b) If all the tiles carried only strings over $\{1\}$ (singleton alphabet), then the problem would be decidable. ☐

(c) The PCP solver included within Jove is, in principle, an algorithm ☐

(d) PCP is a good starting point for proving results such as Grammar Ambiguity being not recursive. For this, we reduce PCP to Grammar Ambiguity. ☐

18. In 1975, Pratt established the result "PRIMES are in NP." In 2002, Agrawal, Kayal, and Saxena established the result "PRIMES are in P."

(a) We now have a P-time algorithm that checks for primality ☐

(b) We can now factor numbers into their prime factors in P-time ☐

(c) Checking whether a given number is Prime in P-time is not the same as factoring numbers in P-time (which is still open) ☐

(d) We have now the result P=NP ☐

6

19. Consider BDDs over variables x and y for the Boolean functions *and*, *or*, and *xor*. The following are true:

    (a) The BDD for *or* will have only one path to the "1" node starting from the root. ☐
    (b) The BDD for *xor* will have two paths to the "1" node starting from the root. ☐
    (c) The BDD for *and* will have one path to the "1" node starting from the root. ☐
    (d) BDDs always have a polynomial number of nodes in them, even though Boolean satisfiability checking is NP-complete. ☐

20. (Part-1): If a Deterministic Turing Machine M is programmed to check if two strings w_1 and w_2 (both of length N) are equal by zig-zagging along the string, the machine M will take **this much runtime** in Big-O notation. (Part-2): A TM that writes into $O(N^3)$ cells at least once must take this much time.

    (a) $O(N)$ for Part-1. ☐
    (b) $O(N^2)$ for Part-1. ☐
    (c) At least $O(N^3)$ time for Part-2. ☐
    (d) Can be lower than $O(N^3)$ time for Part-2. ☐

21. These are impossible or ill-advised:

    (a) Developing an algorithm to check for CFG ambiguity ☐
    (b) Expressing all computable functions using two combinators, namely S and K ☐
    (c) Developing a P-time algorithm to check for Boolean satisfiability ☐
    (d) Avoiding the use of a SAT-solver because SAT is NP-complete ☐

22. New languages L are typically shown NPC by these two steps

    (a) Showing that L is in NP ☐
    (b) Showing that L can be P-time mapping-reduced to an existing NP-complete problem ☐
    (c) Showing that all the NP problems can be mapping-reduced to L ☐
    (d) Showing that some NPC problem can be P-time mapping-reduced to L ☐

23. A new problem L can be shown to be NP-hard. This is not sufficient as an NP-completeness proof for L because:

    (a) L being NP-hard means that L is in NP ☐
    (b) L can be undecidable ☐
    (c) We must additionally show that L is in NP ☐
    (d) (c) *and* the statement "The first language shown NP-hard was the K-Clique problem" ☐

7

24. By "3CNF," "3SAT," and "DNF" (disjunctive normal form; Boolean formulae such as `(abc+def)`), the following are meant:

  (a) When a 3CNF formula is converted to DNF, the size of the resulting formula can be exponential with respect to the input formula ☐

  (b) The satisfiability checking of DNF formulae is also NP-complete ☐

  (c) A 3SAT formula that is not a contradiction (always false) ☐

  (d) When a 3CNF formula is converted to DNF, the size of the resulting formula grows polynomially with respect to the input formula ☐

25. In proving the P-time solvability of 2CNF, we converted "OR"s to implications. In such a construction, `(a+b).(c+!d)` will result in these edges (or implications):

  (a) `!a -> b` ☐

  (b) `!d -> c` ☐

  (c) `!c -> !d` ☐

  (d) `b -> a` ☐

26. Given that 3SAT has been shown to be NP-Complete (NPC), to show that the K-Clique problem is NPC, here are all the additional steps needed:

  (a) Showing that 3SAT is in NP ☐

  (b) Showing that K-Clique is in NP ☐

  (c) Showing that K-Clique can be reduced to 3SAT via a P-time reduction ☐

  (d) Showing that 3SAT can be reduced to K-Clique via a P-time reduction ☐

27. Consider the 3CNF formula `(a+b+c) (a+!b+!c)(!a+!b+!c)` that is being converted into a graph for the purposes of an NP-completeness proof of 3COLOR. The following are true of the graph that is generated from this formula or a small variant of this formula.

  (a) We can conjoin clauses involving `a,b,c` to the original formula to ensure that 3-coloring is blocked. ☐

  (b) The impossibility of 3-coloring implies the impossibility of 4-coloring. ☐

  (c) The impossibility of 3-coloring implies the impossibility of 2-coloring. ☐

  (d) 2-coloring is in P, and a good way to argue that is through depth-first search (DFS). ☐

LONG ANSWER PROBLEMS

You may use Jove to run anything to double-check; but enter your answers into Canvas which is what will be graded.

1. (1 point) Please answer in 2 sentences: describe one topic from CS 3100 that interested or benefited you the most, and why.

2. (9 points, Mapping Reduction from PCP to Grammar Ambiguity) Look at the file `FinalsPrep-with-annotations.pdf` uploaded into the files area.

   The `A` and the `B` sets are (respectively) the top and bottom halves of the PCP tiles.

   The CFG uses three non-terminals, S, S_A, and S_B.

   The alphabet of the CFG includes all the symbols used in the w_i and x_i (tiles) and n extra symbols that do not appear in any of the tiles (called $a_1 \ldots a_n$).

   The productions are as indicated.

   Assume that Figure 15.1 of the book is the source of the A and B sets.

   Assume that the n (four for us) extra symbols are $\{a_1, a_2, a_3, a_4\}$ respectively.

   (a) (6 points) Looking at the PCP solution shown in Figure 15.1, and the hand-drawn hint on why ambiguity arises on Page 3 of `FinalsPrep-with-annotations.pdf`, write out the derivation sequence that uses S_A which gives rise to ambiguity. Your answer must be of this form in the box provided on Canvas; **do not write more than this kind of a derivation sequence**:

   $S \Rightarrow w_p \ S_A \ P \Rightarrow w_p \ w_q \ S_A \ Q \ P \Rightarrow \ldots$

   Your task is to fill in the "p, q" entries by giving actual tile numbers which come from $\{1, 2, 3, 4\}$.

   You must also fill in P, Q, etc., which come from $\{a_1, a_2, a_3, a_4\}$.

   (b) (2 points) What is the role of the a_i's?

   Write in two clear sentences, saying what might go wrong without involving the a_i's. (How can leaving out a_i's make the mapping reduction fail?)

   (c) (1 points) Suppose someone invents a PCP grammar ambiguity checker as a decider. What is the key consequence of such a result? **Answer in one clear sentence**.

3. (9 points, NPC Proof of Clique) Study the NPC proof of Clique from Figure 16.9 of our textbook.

(a) (3 points) Why does this construction serve as a P-time mapping reduction? Please answer in three clear sentences, where **one sentence addresses the P-time aspect**, and the **other two sentences** address the "if and only if" aspects. Please answer in the spaces provided and **do not use more than three sentence total.**

(b) (3 points) Suppose we are given the following CNF formula with three clauses: `(a+b+c).(d+e+f).(g+h+i)` where `a` through `i` are different variables. How many triangles will be formed by the mapping reduction (**please give a number in the space provided**), and why (**please provide one sentence in the space provided**)?

(c) (3 points) Suppose we replace `(g+h+i)` with `(!a+!b+!c)`. How many triangles will be formed by the mapping reduction (**please give a number in the space provided**), and why (**please provide one sentence in the space provided**)?