Computer Science 456/656 Spring 2021 Practice Examination April 8, 2021

The entire examination is 660 points. The actual exam will be shorter.

1. True or False. [5 points each] T = true, F = false, and O = open, meaning that the answer is not known to science at this time.

   i. **T** Let L be the language over $\Sigma = \{a, b\}$ consisting of all strings of the form $a^m b^n$, where $m, n \geq 0$. Then L is a regular language.

   ii. **T** The complement of every regular language is regular.

   iii. **T** The Kleene closure of every context-free language is context-free.

   iv. **F** If a language has an ambiguous context-free grammar, then it is is not accepted by any deterministic push-down automaton.

   v. **F** There is a PDA that accepts all valid C++ programs.

   vi. **T** The intersection of any two regular languages is regular.

   vii. **F** The intersection of any two context-free languages is context-free.

   viii. **T** The set of all base 7 numerals for positive integers n such that $n \ \% \ 3 = 2$ is regular.

   ix. **T** Let L be the language over $\Sigma = \{a, b\}$ consisting of all strings of the form $a^m b^n c^m$, where $m, n \geq 0$. Then L is a context-free language.

   x. **T** Let L be the language over $\Sigma = \{a, b\}$ consisting of all strings of the form $a^m b^n$, where $m \geq n$. Then L is a context-free language.

   xi. **F** The complement of every context-free language is context-free.

   xii. **T** The union of any two context-free languages is context-free.

   xiii. **T** If a language has a context-free grammar, then it is is accepted by some push-down automaton.

   xiv. **F** Every context-free language has an unambiguous context-free grammar.

   xv. **F** Every language that has an unambiguous context-free grammar is accepted by some DPDA.

   xvi. **T** Every deterministic machine is a non-deterministic machine.

   xvii. **T** The language consisting of all base 2 numerals for integer powers of 2 is regular.

   xviii. **F** There is a DPDA that accepts the language of all palindromes over the binary alphabet $\{0, 1\}$.

   xix. **T** The language $\{a^n b^n c^n d^n \mid n \geq 0\}$ is recursive.

   xx. **F** The problem of whether a given context-free grammar generates all strings is decidable.

   xxi. **T** The language $\left\{a^i b^j c^k \mid j \geq i + k\right\}$ is context-free.

xxii. **T** If a language L is undecidable, there is no machine that enumerates L in canonical order.

xxiii. **T** **T** If L is in $\mathcal{RE}$ and also L is in co-$\mathcal{RE}$, then L must be decidable.

xxiv. **F** The context-free grammar equivalence problem is in the class $\mathcal{RE}$.

xxv. **T** The context-free grammar equivalence problem is in the class co-$\mathcal{RE}$.

xxvi. **F** Every bounded function is recursive.

xxvii. **F** If P is a mathematical proposition that can be stated using n binary bits, and P has a proof, then P must have a proof whose length is $O(2^{2^n})$.

xxviii. **T** The complement of every recursive language is recursive.

xxix. **F** The complement of every recursively enumerable language is recursively enumerable.

xxx. **T** Every language which is generated by an general grammar is recursively enumerable.

xxxi. **T** The context-free grammar equivalence problem is undecidable.

xxxii. **F** Given any context-free grammar G and any string $w \in L(G)$, there is always a unique leftmost derivation of w using G.

xxxiii. **F** For any deterministic finite automaton, there is always a unique minimal non-deterministic finite automaton equivalent to it.

xxxiv. **F** Using multi-processors and other advanced technology, it is possible to design a machine which decides the halting problem.

xxxv. **O** The question of whether two regular expressions are equivalent is $\mathcal{NP}$-complete.

xxxvi. **T** The intersection of any context-free language with any regular language is context-free.

xxxvii. **T** The halting problem is recursively enumerable.

xxxviii. **F** The complement of every context-free language is context-free.

xxxix. **F** No language which has an ambiguous context-free grammar can be accepted by a DPDA.

xl. **T** The union of any two context-free languages is context-free.

xli. **F** The question of whether a given Turing Machine halts with empty input is decidable.

xlii. **T** The class of languages accepted by non-deterministic Turing Machines is the same as the class of languages accepted by Turing Machines.

xliii. **T** Let $F(0) = 1$, and let $F(n) = 2^{F(n-1)}$ for $n > 0$. Then F is recursive.

xliv. **T** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine.

xlv. **F** The language of all regular expressions over the binary alphabet is a regular language.

xlvi. **T** There is no computer program that decides whether any two C++ programs are equivalent.

xlvii. **F** Every function that can be mathematically defined is recursive.

xlviii. **T** The language of all binary numerals for multiples of 23 is regular.

xlix. **F** The language of all binary strings which are the binary numerals for prime numbers is context-free.

l. **F** Every bounded function from integers to integers is Turing-computable. (We say that f is *bounded* if there is some B such that $|f(n)| \leq B$ for all n.)

li. **F** The language of all palindromes over $\{0, 1\}$ is inherently ambiguous.

lii. **T** The language $\left\{a^i b^j c^k \,:\, i = j \text{ or } j = k\right\}$ is context-free, but is inherently ambiguous.

liii. **F** Every context-free grammar can be parsed by some deterministic top-down parser.

liv. **T** Every context-free grammar can be parsed by some non-deterministic top-down parser.

lv. **F** Commercially available parsers cannot use the LALR technique, since most modern programming languages are not context-free.

lvi. **F** The diagonal language is $\mathcal{RE}$.

lvii. **T** The diagonal language is co-$\mathcal{RE}$.

lviii. **T** The regular grammar membership problem is in $\mathcal{NC}$.

lix. **T** The context-free grammar membership problem is in $\mathcal{NC}$.

lx. **T** There is a polynomial time reduction of SAT to 3-CNF-SAT.

lxi. **T** There is a language which is neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

lxii. **T** $\mathcal{P}$-SPACE = co-$\mathcal{P}$-SPACE

lxiii. **T** Regular expression equivalence is $\mathcal{P}$-SPACE complete.

2. Fill in the blanks.

   (a) If there is an easy reduction from L_1 to L_2, then L_2 is at least as hard as L_1.

   (b) If a language is accepted by some Turing machine, it is **recursively** enumerable.

   (c) If L_1 is $\mathcal{NP}$ and L_2 is $\mathcal{NP}$-complete, there must be a $\mathcal{P}$-TIME reduction of L_1 to L_2.

   (d) The language HALT is generated by an **unrestricted** grammar.

3. [30 points] State the Church-Turing thesis, and explain (in about 5 lines or less) why it is important.

   If M is any machine, there is a Turing machine equivalent to M.

   This is important because to prove that no machine can perform some computation, it suffices to prove that no Turing machine can perform that computation.

4. [30 points] We give an LALR parser for the context-free grammar with start symbol S and the given productions. (Think of b and n as begin and end, or as { and }.) Walk through a computation of that parser for the input string *bwwaaiaebnn*. The first configuration is $\$_0 : bwwaaiaebnn\$$.

1. $S \to a_2$
2. $S \to i_3 S_4$
3. $S \to i_3 S_4 e_5 S_6$
4. $S \to w_7 S_8$
5. $S \to b_9 L_{10} n_{11}$
6. $L \to L_{10} S_{12}$
7. $L \to \lambda$

	a	i	e	w	b	n	$	S	L
0	s2	s3		s7	s9			1	
1							**halt**		
2	r1	r1	r1	r1	r1	r1	r1		
3	s2	s3		s7	s9			4	
4	r2	r2	s5	r2	r2	r2	r2		
5	s2	s3		s7	s9			6	
6	r3	r3	r3	r3	r3	r3	r3		
7	s2	s3		s7	s9			8	
8	r4	r4	r4	r4	r4	r4	r4		
9	r7	r7		r7	r7	r7			10
10	s2	s3		s7	s9	s11		12	
11	r5	r5	r5	r5	r5	r5	r5		
12	r6	r6	r6	r6	r6	r6			

stack	action	instream	outstream
$\$_0$		$bwwaaiaebnn\$$	
$\$_0 b_9$	$(s9)$	$wwaaiaebnn\$$	
$\$_0 b_9 L_{10}$	$(r7)$	$wwaaiaebnn\$$	7
$\$_0 b_9 L_{10} w_7$	$(s7)$	$waaiaebnn\$$	7
$\$_0 b_9 L_{10} w_7 w_7$	$(s7)$	$aaiaebnn\$$	7
$\$_0 b_9 L_{10} w_7 w_7 a_2$	$(s2)$	$aiaebnn\$$	7
$\$_0 b_9 L_{10} w_7 w_7 S_8$	$(r1)$	$aiaebnn\$$	71
$\$_0 b_9 L_{10} w_7 S_8$	$(r4)$	$aiaebnn\$$	714
$\$_0 b_9 L_{10} S_{12}$	$(r4)$	$aiaebnn\$$	7144
$\$_0 b_9 L_{10}$	$(r6)$	$aiaebnn\$$	71446
$\$_0 b_9 L_{10} a_2$	$(s2)$	$iaebnn\$$	71446
$\$_0 b_9 L_{10} S_{12}$	$(r1)$	$iaebnn\$$	714461
$\$_0 b_9 L_{10}$	$(r6)$	$iaebnn\$$	7144616
$\$_0 b_9 L_{10} i_3$	$(s3)$	$aebnn\$$	7144616
$\$_0 b_9 L_{10} i_3 a_2$	$(s2)$	$ebnn\$$	7144616
$\$_0 b_9 L_{10} i_3 S_4$	$(s2)$	$ebnn\$$	71446161
$\$_0 b_9 L_{10} i_3 S_4 e_5$	$(s5)$	$bnn\$$	71446161
$\$_0 b_9 L_{10} i_3 S_4 e_5 b_9$	$(s9)$	$nn\$$	71446161
$\$_0 b_9 L_{10} i_3 S_4 e_5 b_9 L_{10}$	$(r7)$	$nn\$$	714461617
$\$_0 b_9 L_{10} i_3 S_4 e_5 b_9 L_{10} n_{11}$	$(s11)$	$n\$$	714461617
$\$_0 b_9 L_{10} i_3 S_4 e_5 S_6$	$(r5)$	$n\$$	7144616175
$\$_0 b_9 L_{10} S_{12}$	$(r3)$	$n\$$	71446161753
$\$_0 b_9 L_{10}$	$(r6)$	$n\$$	714461617536
$\$_0 b_9 L_{10} n_{11}$	$(s11)$	$\$$	714461617536
$\$_0 S_1$	$(r5)$	$\$$	7144616175365
	halt		7144616175365

5. [5 points] What class of machines accepts the class of context free languages?

   PDAs.

6. [5 points] What class of machines accepts the class of recursively enumerable languages?

   Turing machines.

7. [10 points] What is the *canonical order* of a language?

   The order where a shorter string is smaller than a longer string, and strings of the same length are ordered alphabetically.

8. [10 points] What does it mean to say that machines M_1 and M_2 are *equivalent*?

If M_1 and M_2 are given the same output, they will give the same output.

9. [10 points] What does it mean to say that a language L is *decidable*?

   Let Σ be the alphabet of L. L is decidable means that there is some machine M such that, given any $w \in \Sigma^*$, M halts, and outputs 1 if $w \in L$ and outputs 0 if $w \notin L$.

10. [10 points] What is a *certificate* or *witness* as used in the definition of the class $\mathcal{NP}$?

    Suppose $L \subseteq \Sigma^*$ is $\mathcal{NP}$. Then there exists a machine V_L which we can call a *verifier* for L, such that:

    (a) If $w \in L$ there exists a string c, called a *certificate* (or *witness*) for w such that if given the input $w\#c$, V returns 1, in time which is polynomial in the length of w.

    (b) If $w \notin L$ and c is any string, given the input $w\#c$ V will return 0 in time which is polynomial in the length of w.

11. [10 points] What does it mean to say that a language is in the class $\mathcal{P}$-SPACE?

    Let $L \in \Sigma^*$. We say L is $\mathcal{P}$-SPACE if there is a Turing machine M and a constant k such that, for any $w \in \Sigma^*$, given input w, M will decide whether $w \in L$ using a computation which uses at most n^k cells of its tape, where $n = |w|$.

12. [20 points] Find a Chomsky normal form grammar equivalent to the context-free grammar given below.

    1. $S \to iS$
    2. $S \to iSeS$
    3. $S \to wS$
    4. $S \to a$

    There are many correct answers. Here is mine.

    1. $S \to a$
    2. $S \to IS$
    3. $S \to WS$
    4. $S \to XY$
    5. $X \to IS$
    6. $Y \to ES$
    7. $I \to i$
    8. $W \to w$
    9. $E \to e$

13. [20 points] Let $\Sigma = \{0, 1\}$, the binary alphabet. We say a string w over Σ is *mostly positive* if $\#_1(w) > \#_0(w)$. Let L be the set of mostly positive strings over Σ.

    Give a context-free grammar for L. **Very hard.**

    There are many correct answers. Here is mine. I'm sure there is an answer with fewer productions.

    1. $S \to T1T$
    2. $T \to 0T1T$
    3. $T \to 1T0T$
    4. $T \to 1T$
    5. $T \to T1$
    6. $T \to \lambda$

14. [10 points] What is a **reduction** of a language L_1 to a language L_2?

    Given $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$, a *reduction* of L_1 to L_2 is a function $R : \Sigma_1^* \to \Sigma_2^*$ such that, for any $w \in \Sigma_1^*$, $R(w) \in L_2$ if and only if $w \in L_1$.

15. (a) [10 points] State the pumping lemma for regular languages accurately. If you have all the right words but in the wrong order, that means you truly do not understand the lemma, and you might get no partial credit at all.

    For any regular language L
    there exists an integer $p > 0$ such that
    for any $w \in L$ such that $|w| \geq p$
    there exist strings x, y, z such that

    1. $w = xyz$
    2. $|xy| \leq p$
    3. $|y| \geq 1$
    4. for any integer $i \geq 0$, $xy^iz \in L$

    (b) [10 points] State the pumping lemma for context-free languages accurately. If you have all the right words but in the wrong order, that means you truly do not understand the lemma, and you might get no partial credit at all.

    For any context-free language L
    there exists an integer $p > 0$ such that
    for any $w \in L$ such that $|w| \geq p$
    there exist strings u, v, x, y, z such that

    1. $w = uvxyz$
    2. $|vxy| \leq p$
    3. $|vy| \geq 1$

4. for any integer $i \geq 0$, $uv^ixy^iz \in L$

16. [20 points] Prove that a recursively enumerable language is accepted by some machine.

Assume that a machine M enumerates L. By the Church-Turing thesis, we can use a program, written in pseudo-code, as the machine that accepts L.

Let $w_1, w_2, \ldots$ be the enumeration of L generated by M.

```
read w
for all i from 1 to ∞
    if w_i = w
        halt and accept.
```

17. [20 points] Prove that a language is recursively enumerable if it is accepted by some machine.

Let M be a machine which accepts $L \subseteq \Sigma^*$. Let w_1, w_2 be the canonical enumerattion of Σ^*. The following program enumerates L.

```
for all t from 1 to ∞
    for all i from 1 to t
        if M accepts w_i within t steps
            write w_i
```

18. [20 points] Prove that any language that can be recursively enumerated in canonical order is recursive.

Let L be a language recursively enumerable in canonical order by a machine M. If L is finite it is recursive. Now assume L is infinite. Let $w_1, w_2, \ldots$ be the enumeration of L generated by M. The following program decides L.

```
read w
for all i from 1 to ∞
    if w_i = w
        halt and accept
    else if w_i > w in the canonical order
        halt and reject
```

19. [20 points] Prove that a recursive language can be recursively enumerated in canonoical order.

Let $L \subseteq \Sigma^*$ be a recursive (decidable) language. Let $w_1, w_2 \ldots$ be the enumeration of Σ^* in canonical order. The following program enumerates L in canonical order.

for all i from 1 to ∞
    if $w_i \in L$ // this condition is decided within finite time
        write w_i

20. [20 points] Give a polynomial time reduction of the subset sum problem to the partition problem.

Let $K, x_1, x_2, \ldots x_n$ be an instance of the subset sum problem. This instance is correct if the sum of some subsequence of $\{x_i\}$ is K. Let $S = \sum_{i=1}^{n} x_i$.

Let $y_1, y_2, \ldots y_{n+2}$ be the instance of the partition problem where $y_{n+1} = K + 1$, $y_{n+2} = S - K + 1$, and $y_i = x_i$ for $i \leq n$. Note that $\sum_{i=1}^{n+2} y_i = 2S + 2$.

If A is solution to the instance of the subset sum problem, namely A is a subsequence of $\{x_i\}$ whose total is K, then $A \cup \{y_{n+2}\}$ has total $S+1$, and hence gives a solution to the instance of the partition problem. Conversely, suppose B is a solution to the partition problem, namely a subsequence of $\{y_i\}$ whose total is $S+1$. Let C be the complement of B, which also has total $S+1$. Without loss of generality, B contains y_{n+2}. Then B cannot contain y_{n+1}, because then the total of B would exceed $S+1$. Then $B \backslash \{y_{n+2}\}$ is a subsequence of $\{x_i\}$ whose total is K, giving us a solution to the instance of the subset sum problem.

21. [20 points] Give a polynomial time reduction of 3-CNF-SAT to the independent set problem.

Let E be a Boolean expression in 3-CNF form, that is, $E = C_1 C_2 ... C_k$, where each clause is the dijunction of three terms, each of which is a variable or the negation of a variable. Write $C_i = t_{i,1} + t_{i,2} + t_{i,3}$ and each $t_{i,j}$ is of the form x_ℓ or $!x_\ell$ for some variable x_ℓ. We define a graph G_E consisting of $3k$ vertices and a number of edges, and show that G_E has an independent k-set if and only if E has a satisfying assignment.

The vertices of G_E are $V = \{v_{i,j}\}$ for $1 \leq i \leq k$ and $1 \leq j \leq 3$. Each vertex corresponds to one term of a clause of E. We define $Q_i = \{v_{i,1}, v_{i,2}, v_{i,3}\}$, the vertices corresponding to the clause C_i. We connect those three vertices with "short" edges so that Q_i is a clique.

We connect vertices with "long" edges if they are in different cliques, and if their corresponding terms contradict each other, meaning that one of the terms is x_ℓ for some variable x_ℓ, and and the other is $!x_\ell$.

Suppose E has a satisfying assignment, namely an assignment of each variable to a Boolean value, such that each clause evaluates to true. For each clause C_i, select one term of that clause which is true under the assignment, and call it the *witness* for that clause. A witness can be either a variable or the negation of a variable. Now let I be the set of vertices of G_E which correspond to witnesses. Since there is one witness for each of the k clauses, I has cardinality k. We now show that I is independent. There can be no short edge between to members of I, since there is exactly one in each Q_i. There can also be no long edge, since the two ends of any long edge contradict each other and hence cannot both correspond to witnesses.

Conversely, suppose G has a k-independent set I. Since I is independent, it cannot have more than one member in each clique Q_i. Since there are k cliques, I must have one member in each clique. Since I is

independent, it cannot have both vertices at the ends of any long edge, which means that the vertices corresponding to the members of I do not contradict each other.

We define an assignment of E as follows. If a member of I corresponds to a term which is a variable x_{ell}, then we assign that variable true. If a member of I corresponds to a term which is the negation of a variable $!x_\ell$, we assign that variable false. Any variable not yet assigned is given an arbitrary Boolean value. Each clause has one term which corresponds to a member of I and hence is true. Thus, the assignment satisfies E.