University of Nevada, Las Vegas Computer Science 456/656 Spring 2026

Answers to Practice Problems for Final Examination on May 11

I didn't even try to remove duplicates. Since this will not be graded, just ignore the duplicate questions.

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

   This proof is in the handout fourpfs.pdf.

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

   This proof is in the handout fourpfs.pdf.

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

   This proof is in the handout fourpfs.pdf.

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

   This proof is in the handout fourpfs.pdf.

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. $S \to aSbS$
   2. $S \to \lambda$

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

   Proof: Assume L is regular. Let p be the pumping length of L. Let $w = a^p b^p \in L$. Then there exist strings x, y, z which satisfy the four concluding statements of the pumping lemma. By conclusions 1. and 2. $w = xyz$ and $xy \leq p$ Thus xy is a prefix of $w = xyz$ of length p, which implies that y is a substring of a^p. Write $y = a^k$. By conclusion 3., $k > 0$, and thus $xz = a^{p-k}b^p c^p$. By conclusion 4., $xz \in L$, which contradicts the definition of L. We conclude that L is not regular. □

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

   That proof is too hard for an exam. But I expect you to know that L_D is regular.

6. We know 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? Think!

   Regular languages.

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?)

   Proof: By dynamic programming. The proof is inspired by the CYK algorithm. □

1. True/False/Open

   (i) **T** A minimal DFA equivalent to a given NFA with 3 states can have no more than 7 states which are not dead.

   (ii) **F** Every subset of a regular language is regular.

   (iii) **F** Let L be the language over $\{a, b, c\}$ consisting of all strings which have more a's than b's and more b's than c's. There is some PDA that accepts L.

   (iv) **T** The intersection of any regular language with any context-free language is context-free.

   (v) **F** The intersection of any two context-free languages is context-free.

   (vi) **T** Every language accepted by a non-deterministic machine is accepted by some deterministic machine.

   (vii) **T** The language $\{a^n b^n c^n : n \text{ is prime}\}$ is decidable.

   (viii) **T** If a language L is EXP–SPACE complete, L must be decidable.

   (ix) **O** $\mathcal{NC} = \mathcal{P}$.

   (x) **O** $\mathcal{P} = \mathcal{NP}$.

   (xi) **F** EXP-TIME = $\mathcal{P}$-TIME. (By the time hierarchy theorem.)

   (xii) **O** The Boolean Circuit Problem is in $\mathcal{NC}$.

   (xiii) **O** 3-SAT is $\mathcal{P}$-TIME.

   (xiv) **T** Addition of binary numerals is in $\mathcal{NC}$.

   (xv) **F** Every language generated by a general grammar is recursive.

   (xvi) **T** Every language generated by a general grammar is recursively enumerable.

(xvii) **T** The problem of whether two given context-free grammars are equivalent is co-$\mathcal{RE}$.

(xviii) **T** The language of all fractions (using base 10 numeration) whose values are less than π is decidable. (Recall that a fraction is a string consisting of two numerals separated by "/".

(xix) **T** If L is $\mathcal{RE}$ and co-$\mathcal{RE}$, then L must be decidable.

(xx) **F** For every real number x, there exists a machine that runs forever and outputs the string of decimal digits of x.

(xxi) **T** There exists a mathematical proposition that can be neither proved nor disproved.

(xxii) **T** If a Boolean expression is satisfiable, there is a $\mathcal{P}$–TIME proof that it's satisfiable.

(xxiii) Duplicate

(xxiv) **F** Every subset of any recursively enumerable set is recursively enumerable.

(xxv) **T** The binary numeral factorization problem is co-$\mathcal{NP}$.

2. Every language, or problem, falls into exactly one of these categories.

A Known to be $\mathcal{NC}$.
B Known to be $\mathcal{P}$–TIME, but not known to be $\mathcal{NC}$.
C Known to be $\mathcal{NP}$, but not known to be $\mathcal{P}$–TIME and not known to be $\mathcal{NP}$–complete.
D Known to be $\mathcal{NP}$–complete.
E Known to be $\mathcal{P}$–SPACE but not known to be $\mathcal{NP}$
F Known to be EXP–TIME but not nown to be $\mathcal{P}$–SPACE.
G Known to be EXP–SPACE but not nown to be EXP–TIME.
H Known to be decidable, but not nown to be EXP–SPACE.
K $\mathcal{RE}$ but not decidable.
L co-$\mathcal{RE}$ but not decidable.
M Neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

For each of the languages, write a letter indicating the correct category.

(i) **D** All satisfiable Boolean expressions.

(ii) **C** All binary numerals for composite integers.

(iii) **D** SAT.

(iv) **D** 3-SAT.

(v) **B** 2-SAT.

(vi) **D** The Independent Set problem.

(vii) **D** All satisfiable Boolean expressions.

(viii) **K** All C++ programs which halt with no input.

(ix) **A** All base 10 numerals for perfect squares.

(x) **L** All context-free grammars that generate the Dyck language.

(xi) **E** All configurations of RUSH HOUR from which it's possible to win.

3. Give a definition of each term.

   (i) Accept. (That is, what does it mean for a machine to accept a language.)

   M accepts a language L if, for any $w \in L$, the execution of M with input $w \in L$ halts in an accepting state.

   (ii) Decide. (That is, what does it mean for a machine to decide a language.)

   M decides a language L if, for any $w \in L$, the execution of M with any input $w \in L$ halts in an accepting state, and M halts in a rejecting state if the input is $w \notin L$.

   (iii) Canonical order of a language L.

   For strings u and v, we say $u < v$ in canonical order if either $|u| < |v|$ or $|u| = |v|$ and $u < v$ in lexical order.

1. Find a minimal DFA equivalent to the NFA shown in Figure 1.

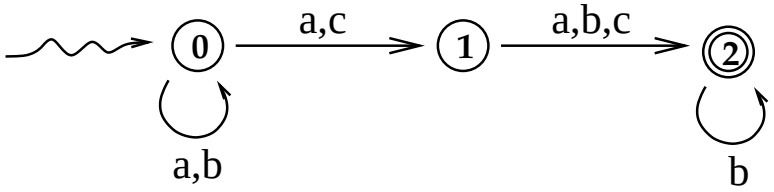
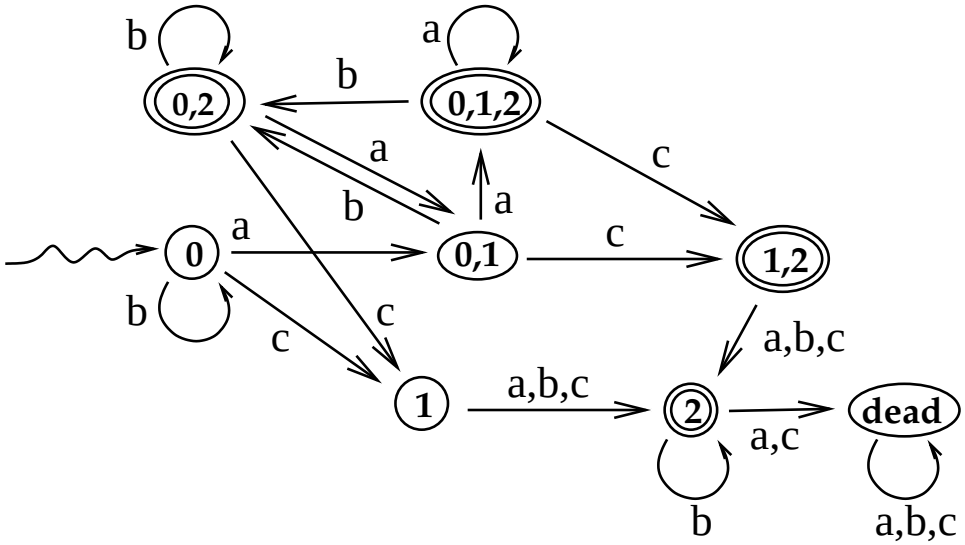


Figure 1



2. Give a regular grammar with **no more than three variables** for the language accepted by the machine in Figure 1.

We identify state 0 with S, state 1 with A, and state 2 with B. Here is the grammar.

1. $S \to bS$
2. $S \to aS$
3. $S \to aA$
4. $S \to cA$
5. $A \to aB$
6. $A \to bB$
7. $A \to cB$
8. $B \to bB$
9. $B \to \lambda$

3. Give a regular expression for the language accepted by the machine in Figure 1

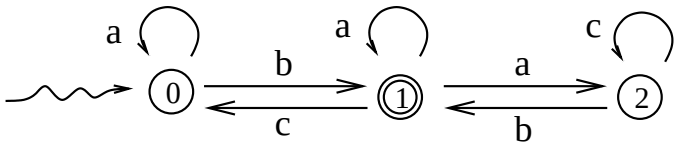


Figure 1: NFA for Problem 3.

The regular expression $a^*b(ca^*b + a + ac * b)^*$

What class of languages does each of these machine classes accept?

(a) Deterministic finite automaton. **Regular languages**

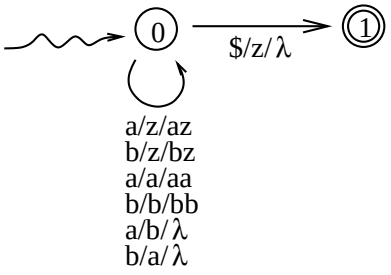
(b) Non-deterministic finite automata. **Regular languages.**

(c) Push-down automata. **Context-free languiages.**

(d) Turing Machines. **Recursively enumerable languages.**

4. Let $L = \{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$, which is generated by the following context-free grammar. Draw a DPDA which accepts L.

1. $S \to aSbS$
2. $S \to bSaS$
3. $S \to \lambda$



5. The grammar below is an ambiguous CF grammar and is parsed by the LALR parser whose ACTION and GOTO tables are shown here. Write a computation of the parser for the input string $iiwaea$.

1. $S \to i_2 S_3$
2. $S \to i_2 S_3 e_4 S_5$
3. $S \to w_6 S_7$
4. $S \to a_8$

	a	i	e	w	$	S
0	s8	s2		s6		1
1					halt	
2	s8	s2		s6		3
3			s4		r1	
4	s8	s2		s6		5
5			r2		r2	
6	s8	s2		s6		7
7			r3		r3	
8			r4		r4	

$\$_0$	$iiwaea\$$		
$\$_0 i_2$	$iwaea\$$		$s2$
$\$_0 i_2 i_2$	$waea\$$		$s2$
$\$_0 i_2 i_2 w_6$	$aea\$$		$s6$
$\$_0 i_2 i_2 w_6 a_8$	$ea\$$		$s6$
$\$_0 i_2 i_2 w_6 S_7$	$ea\$$	4	$r4$
$\$_0 i_2 i_2 S_3$	$ea\$$	43	$r3$
$\$_0 i_2 i_2 S_3 e_4$	$a\$$	43	$s4$
$\$_0 i_2 i_2 S_3 e_4 a_8$	$\$$	43	$s8$
$\$_0 i_2 i_2 S_3 e_4 E_5$	$\$$	434	$r4$
$\$_0 i_2 S_3$	$\$$	4342	$r2$
$\$_0 S_1$	$\$$	43421	$r1$

HALT

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known science at this time. In the questions below, $\mathcal{P}$ and $\mathcal{NP}$ denote $\mathcal{P}$-TIME and $\mathcal{NP}$-TIME, respectively.

   (i) Duplicate

   (ii) **T** The language $\{a^n b^n \mid n \geq 0\}$ is context-free.

   (iii) **F** The language $\{a^n b^n c^n \mid n \geq 0\}$ is context-free.

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

   (v) Duplicate

   (vi) Duplicate

   (vii) **T** If L is a context-free language over an alphabet with just one symbol, then L is regular.

   (viii) **T** The set of strings that your high school algebra teacher would accept as legitimate expressions is a context-free language.

   (ix) Duplicate

   (x) **T** The problem of whether a given string is generated by a given context-free grammar is decidable.

   (xi) **F** Every language generated by an unambiguous context-free grammar is accepted by some DPDA.

   (xii) **T** The language $\{a^n b^n c^n \mid n \geq 0\}$ is in the class $\mathcal{P}$-TIME.

   (xiii) **O** There exists a polynomial time algorithm which finds the factors of any positive integer, where the input is given as a binary numeral.

   (xiv) **F** The intersection of any two undecidable languages is undecidable.

   (xv) **T** Every $\mathcal{NP}$ language is decidable.

   (xvi) **T** The intersection of two $\mathcal{NP}$ languages must be $\mathcal{NP}$.

(xvii) **O** There exists a $\mathcal{P}$-TIME algorithm which finds a maximum independent set in any graph G.

(xviii) Duplicate

(xix) Duplicate

(xx) **O** $\mathcal{NP} = \mathcal{P}$-SPACE

(xxi) Duplicate

(xxii) **F** EXP-SPACE = $\mathcal{P}$-SPACE.

By the space hierarchy theorem.

(xxiii) **T** The traveling salesman problem (TSP) is $\mathcal{NP}$-complete.

(xxiv) **T** The knapsack problem is $\mathcal{NP}$-complete.

(xxv) **T** The language consisting of all satisfiable Boolean expressions is $\mathcal{NP}$-complete.

(xxvi) Duplicate

(i) Duplicate

(ii) **T** 2-SAT is $\mathcal{P}$-TIME.

(iii) Duplicate

(iv) **T** Primality, using binary numerals, is $\mathcal{P}$-TIME.

(v) **T** There is a $\mathcal{P}$-TIME reduction of the halting problem to 3-SAT.

(vi) **T** Every context-free language is in $\mathcal{P}$.

(vii) **T** Every context-free language is in $\mathcal{NC}$.

(viii) Duplicate

(ix) **F** The problem of whether two given context-free grammars generate the same language is decidable.

(x) Duplicate

(xi) **T** For any two languages L_1 and L_2, if L_1 is undecidable and there is a recursive reduction of L_1 to L_2, then L_2 must be undecidable.

(xii) **F** If P is a mathematical proposition that can be written using a string of length n, and P has a proof, then P must have a proof whose length is $O(2^{2^n})$.

(xiii) **T** If L is any $\mathcal{NP}$ language, there must be a $\mathcal{P}$–TIME reduction of L to the partition problem.

(xiv) **F** Every bounded function is recursive.

(xv) Duplicate

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

(xvii) **T** Every language is enumerable.

(xviii) **F** If a language L is undecidable, then there can be no machine that enumerates L.

(xix) **T** There is a non-recursive function which grows faster than any recursive function.

(xx) _____ There exists a machine that runs forever and outputs the string of decimal digits of π (the well-known ratio of the circumference of a circle to its diameter).

(xxi) **F** If a language L is undecidable, then there can be no machine that

(xxii) Duplicate

(xxiii) **O Rush Hour**, the puzzle sold in game stores everywhere, generalized to a board of arbitrary size, is $\mathcal{NP}$–complete.

(xxiv) **T** There is a polynomial time algorithm which determines whether any two regular expressions are equivalent.

(xxv) **T** If two regular expressions are equivalent, there is a polynomial time proof that they are equivalent.

(xxvi) Duplicate

(xxvii) Duplicate

(xxviii) Duplicate

(xxix) **T** If L is a context-free language which contains the empty string, then $L\backslash\{\lambda\}$ must be context-free.

(xxx) **T** The computer language C++ has Turing power.

(xxxi) Duplicate

(xxxii) Duplicate

(xxxiii) **T** If an abstract Pascal machine can perform a computation in polynomial time, there must be some Turing machine that can perform the same computation in polynomial time.

(xxxiv) Duplicate

1. True/False/Open

   (i) **F** Every subset of a regular language is decidable.

   (ii) **T** The intersection of any two $\mathcal{NP}$ languages is $\mathcal{NP}$. **Think!**

   (iii) **T** Every language accepted by a non-deterministic machine is accepted by some deterministic machine.

   (iv) Duplicate

   (v) **T** $\mathcal{P} = \mathcal{NP}$.

   (vi) Duplicate

(vii) **O** The independent set problem is $\mathcal{P}$-TIME.

(viii) **T** If S is a recursive set of positive integers, then $\sum_{n \in S} 2^{-n}$ must be a recursive real number.

(ix) **T** Multiplication of matrices with binary numeral entries is $\mathcal{NC}$.

(x) **T** Equivalence of regular expressions is decidable.

(xi) **T** Every recursively enumerable language is generated by a general grammar.

(xii) **T** Equivalence of context-free grammars is co–$\mathcal{RE}$.

(xiii) **T** The language consisting of all fractions whose values are less than the natural logarithm of 5.0 is recursive.

(xiv) **T** If L is in $\mathcal{RE}$ and also co–$\mathcal{RE}$, then L must be decidable.

(xv) Duplicate

(xvi) **T** Every sliding block problem is $\mathcal{P}$–SPACE.

(xvii) **F** There are uncountably many co–$\mathcal{RE}$ languages over a given alphabet.

(xviii) **T** If L is any $\mathcal{P}$–TIME language, there is an $\mathcal{NC}$ reduction of L to CVP, the Boolean circuit problem.

(xix) Duplicate

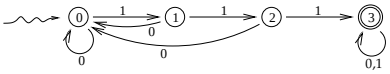
(xx) **T** Every finite language is regular.

(xxi) **T** If L is a $\mathcal{P}$–TIME language, there is a Turing Machine which decides L in polynomial time.

(xxii) **T** If anyone ever finds a polynomial time algorithm for any $\mathcal{NP}$–complete language, then $\mathcal{P} = \mathcal{NP}$.

(xxiii) **T** RSA encryption is believed to be secure because it is believed that the factorization problem for integers is very hard.

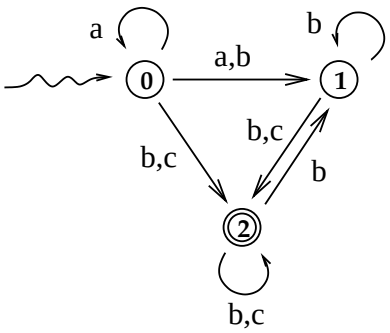
2. Find a DFA with at most 4 states which accepts the language of binary strings which contain the substring 111.

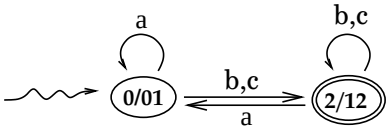


3. Let L be the language of all binary numerals for positive multiples of either 3 or 4, where leading zeros are not permitted. That is, $L = \{11, 100, 110, 1000, 1001, 1100, \ldots\}$. Find an NFA with 8 states which accepts L. There is also a DFA with 12 states which accepts L.

   This problem is too lengthy for an exam.

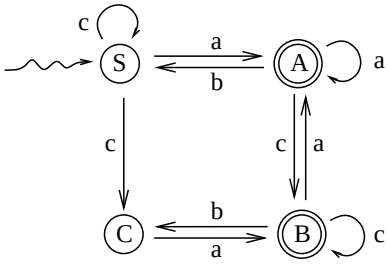
4. Construct a minimal DFA equivalent to the NFA shown below.





5. Find an NFA which accepts the language generated by this grammar.

$S \to aA|cS|cC$
$A \to aA|bS|cB|\lambda$
$B \to aA|cB|bC|\lambda$
$C \to aB$

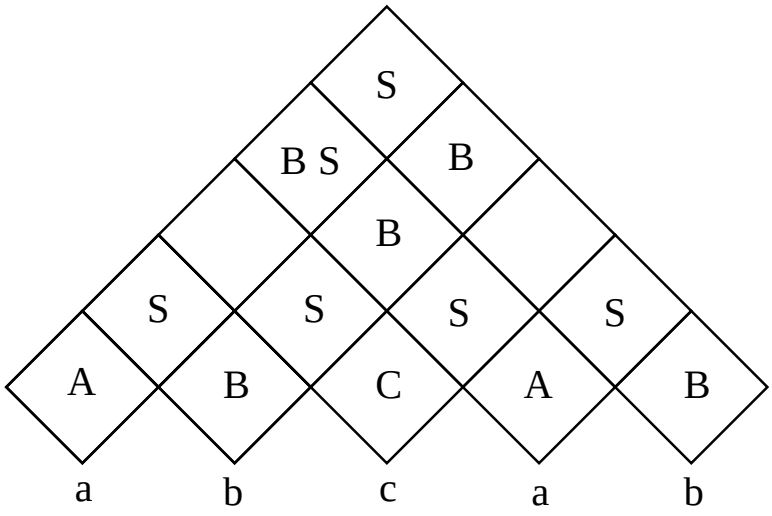


6. Duplicate

7. Use the CYK algorithm to decide whether *abcab* is generated by the CNF grammar:

$S \to AB \,|\, BC \,|CA$
$A \to a$
$B \to SA \,|\, SS \,|\, b$
$C \to c$

by filling in the matrix.



8. The LALR parser given for this grammar:
   1. $E \to E -_2 E_3$
   2. $E \to E *_4 E_5$
   3. $E \to x_6$

   contains errors, meaning that it might parse a string in a manner that would be considered incorrect by your programming instructor. Find those errors and correct them.

   The erros were entry $(5, -)$ and $(5, *)$.

	x	$-$	$*$	\$	E
0	$s6$				1
1		$s2$	$s4$	*halt*	
2	$s6$				3
3		$r1$	$r1$	$r1$	
4	$s6$				5
5		$r2$	$r4$	$r2$	
6		$r3$	$r3$	$r3$	

1. Prove that the grammar given in Problem 8 is ambiguous by giving two different leftmost derivations for some string. (If you simply give two different parse trees, you have not answered the question.)

   $E \Rightarrow E - E \Rightarrow x - E \Rightarrow x - E * E \Rightarrow x - x * E \Rightarrow x - x * x$

   $E \Rightarrow E * E \Rightarrow E - E * E \Rightarrow x - E * E \Rightarrow x - x * E \Rightarrow x - x * x$

   The first derivation is consistent with the usual precedence of operators, while the second one is not.

2. State the pumping lemma for regular languages.

   For any regular language L
   There exists an integer p such that
   For any $w \in L$ where $|w| \geq p$
   There exist string x, y, and z such that $w = xyz$, $|xy| \leq p$, $y \neq \lambda$, and
   For any $i \geq 0$ $xy^iz \in L$.

3. Give the verifier-certificate definition of the class $\mathcal{NP}$.

   If L is $\mathcal{NP}$, there is a number k and a machine M, called the verifier, such tha for any $w \in L$ there is a string c, called the certificate of w, such that M accepts the string (w, c) in $O(|w|^k)$ time, and such that, if $w \notin L$, there is no certificate of w, meaning that M does not accept (w, c) for any string c.

4. What is the importance nowadays of $\mathcal{NC}$?

   There are more and more parallel machines. We would like to run algorithms faster by partitioning the work among processors. Is it possible to do that with any sequenctial algorithm? If $\mathcal{P} = \mathcal{NC}$, then the answer is yes, but that queston still open.

5. What complexity class contains sliding block problems?

   $\mathcal{P}$–SPACE–complete

6. Give a polynomial time reduction of the subset sum problem to the partition problem.

   That reduction is given in the handout npAll.pdf.

7. Duplicate

8. Duplicate

9. Duplicate

10. The grammar below is an ambiguous CF grammar with start symbol E, and is parsed by the LALR parser whose ACTION and GOTO tables are shown here. The ACTION table is missing actions for the second column, when the next input symbol is the "minus" sign. Fill it in. Remember the C++ precedence of operators. (Hint: the column has seven different actions: s2, s4, r1, r2, r3, r4, and r5, some more than once, and has no blank spaces.)

1. $E \to E -_2 E_3$
2. $E \to -_4 E_5$
3. $E \to E *_6 E_7$
4. $E \to ({}_8 E_9)_{10}$
5. $E \to x_{11}$

	x	$-$	$*$	(	)	\$	S
0	$s11$	$s4$		$s8$			1
1		$s2$	$s6$			halt	
2	$s11$	$s4$	$s4$	$s8$			3
3		$r1$	$s6$		$r1$	$r1$	
4	$s11$	$s4$		$s8$	$r4$		5
5		$r2$	$r2$		$r2$	$r2$	
6	$s11$	$s4$		$s8$			7
7		$r3$	$r3$		$r3$	$r3$	
8	$s11$	$s4$		$s8$			9
9		$s2$	$s6$		$s10$		
10		$r4$	$r4$		$r4$	$r4$	
11		$r5$	$r5$		$r5$	$r5$	

11. Prove that any decidable language can be enumerated in canonical order by some machine.

    This proof is in the handout fourpfs.pdf.

12. Give a polynomial time reduction of 3-SAT to to the independent set problem.

    This reduction is in the handout npAll.pdf.

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known science at this time. In the questions below, $\mathcal{P}$ and $\mathcal{NP}$ denote $\mathcal{P}$-TIME and $\mathcal{NP}$-TIME, respectively.

   (i) **F** There is some PDA that accepts $\left\{w \in \{a,b,c\}^* : \#_a(w) > \#_b(w) > \#_c(w)\right\}$, that is, the language over $\{a,b,c\}$ consisting of all strings which have more a's than b's and more b's than c's.

   (ii) Duplicate

   (iii) **T** Every language accepted by a non-deterministic machine is accepted by some deterministic machine.

   (iv) **F** The set of palindromes over $\{a,b\}$ is accepted by some DPDA.

   (v) **T** The language $\{a^n b^n c^n \mid n \geq 0\}$ is in the class $\mathcal{NC}$.

   (vi) Duplicate

   (vii) **F** Every problem that can be mathematically defined has an algorithmic solution.

   (viii) **T** The complement of any undecidable language is undecidable.

   (ix) **O** $\mathcal{NC} = \mathcal{P}$.

   (x) **O** $\mathcal{P} = \mathcal{NP}$.

(xi) **T** The Boolean Circuit Problem (also known as the CVP problem) is in $\mathcal{P}$.

(xii) Duplicate

(xiii) **T** co-$\mathcal{P} = \mathcal{P}$.

(xiv) Duplicate

(xv) Duplicate

(xvi) Duplicate

(xvii) **F** Every language generated by an unrestricted (general) grammar is recursive.

(xviii) **O** If L is $\mathcal{NP}$ and also co-$\mathcal{NP}$, then L must be $\mathcal{P}$–TIME.

(xix) Duplicate

(xx) **T** There is a mathematical proposition that is true but cannot be proved true.

(xxi) Duplicate

(xxii) Duplicate %**T** %There is a Turing machine which, when turned on, runs forever, %writing the decimal expansion of π (the well-known ratio of the %circumference of a circle to its diameter).

(xxiii) **T** The binary integer factorization problem is co-$\mathcal{NP}$.

(xxiv) **T** If L is $\mathcal{NP}$, there is a polynomial time reduction of L to the subset sum problem.

(xxv) Duplicate

(xxvi) **T** The intersection of any two co-$\mathcal{NP}$ languages is co-$\mathcal{NP}$.

(xxvii) **T** The intersection of any two co-$\mathcal{RE}$ languages is co-$\mathcal{RE}$.

(xxviii) **T** Multiplication of matrices with binary numeral entries is $\mathcal{NC}$.

(xxix) **T** Every recursively enumerable language is generated by an unrestricted (general) grammar.

(xxx) **T** Equivalence of context-free grammars is co–$\mathcal{RE}$.

(xxxi) **F** The language of all true mathematical statements is recursively enumerable.

(xxxii) **T** The language of all **provably** true mathematical statements is recursively enumerable.

(xxxiii) **T** There are uncountably many undecidable languages over the binary alphabet.

(xxxiv) **T** If there exists a polynomial time algorithm for any $\mathcal{NP}$–complete problem, then $\mathcal{P} = \mathcal{NP}$.

(xxxv) **T** RSA encryption is accepted as secure by most experts, because they believe that the factorization problem for binary numerals is very hard.

(xxxvi) **F** The language of all $\langle G_1 \rangle \langle G_2 \rangle$ such that G_1 and G_2 are CF grammars which are **not** equivalent is co-$\mathcal{RE}$.

(xxxvii) **T** A real number x is recursive if and only if the set of fractions whose values are greater than x is recursive (decidable).

(xxxviii) Duplicate

(xxxix) **T** If a Boolean expression is satisfiable, there is a $\mathcal{P}$–TIME proof that it's satisfiable.

(xl) **O** If there is a solution to a given instance of any sliding block problem, there must be a solution of polynomial length.

(xli) **T** If the Boolean circut problem (CVP) is $\mathcal{NC}$, then $\mathcal{P} = \mathcal{NC}$.

2. Give a context-sensitive grammar for $\{a^n b^n c^n : n > 0\}$

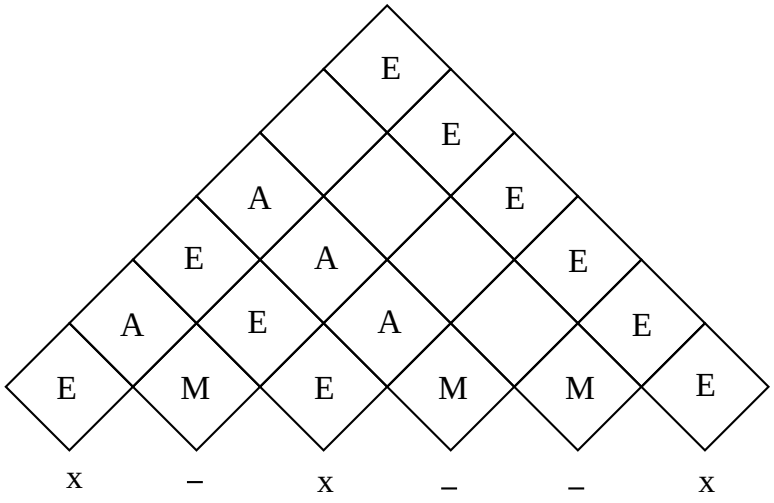
   1. $S \to abc$
   2. $S \to aAabc$
   3. $Aa \to aAAa$
   4. $Aa \to aA$
   5. $Ab \to bB$
   6. $Bb \to bB$
   7. $Bc \to bcc$

3. Using the context-sensitive grammar you wrote above, give a derivation of the string *aaabbbccc*.

   $\underline{S} \Rightarrow a\underline{Aa}bc \Rightarrow aaA\underline{Aa}bc \Rightarrow aaAa\underline{Ab}c \Rightarrow aaAab\underline{Bc} \Rightarrow aa\underline{Aa}bbcc \Rightarrow aaa\underline{Ab}bcc \Rightarrow aaab\underline{Bb}cc$
   $\Rightarrow aaabb\underline{Bc}c \Rightarrow aaabbbccc$

4. Use the CYK algorithm to decide whether $x - x - -x$ is generated by the CNF grammar below, by filling in the matrix.

   $E \to ME$
   $A \to EM$
   $E \to AE$
   $M \to -$
   $E \to x$



5. What is the importance nowadays of $\mathcal{NC}$?

6. Label each of the following sets as countable or uncountable.

   (i) **countable** The set of integers.

(ii) **countable** The set of rational numbers.

(iii) **uncountable** The set of real numbers.

(iv) **uncountable** The set of binary languages.

(v) **countable** The set of co-RE binary languages.

(vi) **uncountable** The set of undecidable binary languages.

(vii) **uncountable** The set of functions from integers to integers.

(viii) **countable** The set of recursive real numbers.

(ix) **countable** The set of $\mathcal{P}$–SPACE languages over the binary alphabet.

(x) **countable** The set of functions from the integers to the binary alphabet $\{0, 1\}$.

1. Duplicate

2. Consider the CF grammar below. The ACTION and GOTO tables are given below, except that six actions are missing, indicated by question marks. Fill in the missing actions (below the question marks). The actions of your table must be consistent with the precedence of operators in C++.

1. $E \to E -_2 E_3$
2. $E \to -_4 E_5$
3. $E \to E *_6 E_7$
4. $E \to x_8$

	x	$-$	$*$	\$	E
0	$s8$	$s4$			1
1		$s2$	$s6$	HALT	
2	$s8$	$s4$			3
3		$r1$	$s6$	$r1$	
4	$s8$	$s4$			5
5		$r2$	$r2$	$r2$	
6	$s8$	$s4$			7
7		$r3$	$r3$	$r3$	
8		$r4$	$r4$	$r4$	

3. Prove, by induction, that $\sum_{i=1}^{n} n^2 = \dfrac{n(n+1)(2n+1)}{6}$

   Proof: We first prove the base case, that the statement is true for $n = 1$. In that case, the left side is 1, and the right side is $\dfrac{1*2*3}{6} = 1$. Thus, the base case holds.

   Next we prove the inductive step. Assume that if the statement holds for $n-1$. We need to prove that it holds for n.

$$\begin{aligned}
\sum_{i=1}^{n} n^2 &= \sum_{i=1}^{n-1} i^2 + n^2 \\
&= \frac{(n-1)n(2n-1)}{6} + n^2 \text{ by the inductive hypothesis} \\
&= \frac{2n^3 - 3n^2 + n + 6n^2}{6} \\
&= \frac{2n^3 + 3n^2 + n}{6} \\
&= \frac{n(n+1)(2n+1)}{6}
\end{aligned}$$

   and we are done. □

4. Prove, by contradiction, that $\sqrt{3}$ is irrational.

   Proof: Assume that $\sqrt{3} = pq$ where the fraction is reduced to the lowest terms, that is, $\gcd(p, q) = 1$.

$$\begin{aligned}
3 &= \frac{p^3}{q^3} \text{ by squaring both sides} \\
3q^2 &= p^2
\end{aligned}$$

   Thus p^2 is a multiple of 3, which implies that p is a multiple of 3. say $p = 3k$. Then $p^2 = 9k$. $3q^2 = 9k$, hence $q^2 = 3k$. Thus q^2 is a multiple of 3, and thus q is a multiple of 3. Thus, p and q have a common divisor of 3, which implies that $\gcd(p, q) \geq 3$, contradiction. Thus $\sqrt{3}$ is irrational. □

5. Prove that the halting problem is undecidable.

   The proof is in the handout haltund.pdf