University of Nevada, Las Vegas Computer Science 456/656 Fall 2025

Answers to Assignment 6: Due November 8, 2025

$\mathcal{P}$ means $\mathcal{P}$–TIME.

1. True/False. If the answer is not known to science at this time, enter "O" for Open.

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

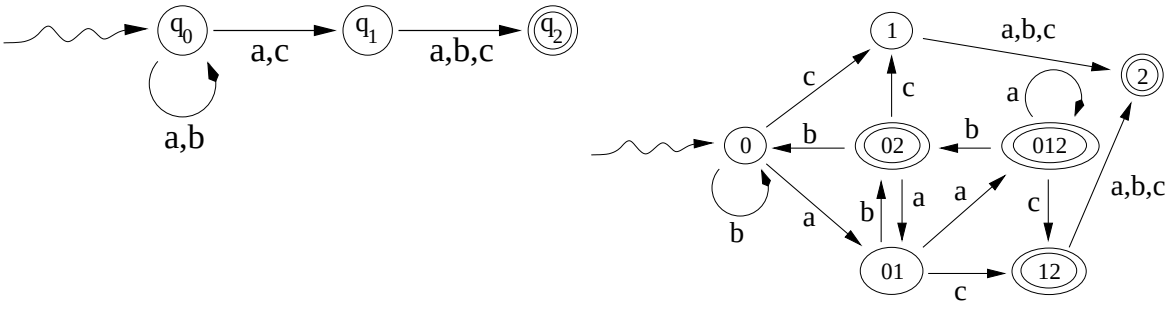
   (b) **T** Given any $\mathcal{P}$–TIME problem P, there is a $\mathcal{NC}$ reduction of P to the circuit value problem.

   (c) **T** Every context-free languags is $\mathcal{NC}$.

   (d) **F** If a language L is generated by a general grammar, L must be decidable.

   (e) **T** Every undecidable language is $\mathcal{NP}$-hard.

2. Draw a DFA equivalent to the following NFA.



3. Give a definition of the class $\mathcal{P}$-complete, and name a language in that class.

   A language L is $\mathcal{P}$–complete if every $\mathcal{P}$–TIME language can be reduced to L using a reduction that can be computed in polylogarithmic time using polynomially many processors. The circuit value problem, CVP, is known to be $\mathcal{P}$–complete.

4. Every language, or problem, falls into exactly one of these categories. For each of these languages or problems, write a letter indicating the correct category. You will need to search the internet for some of these.

   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 known to be $\mathcal{P}$–SPACE.
   G Known to be EXP–SPACE but not known to be EXP–TIME.
   H Known to be decidable, but not known to be EXP–SPACE.
   I $\mathcal{RE}$ but not decidable.
   K co-$\mathcal{RE}$ but not decidable.
   L Neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

(i) **B** The circuit value problem (CVP).

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

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

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

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

(vi) **B** Binary numerals for composite integers. (Composite means not prime.)

(vii) **A** Decimal numerals for positive multiples of seven, such as 7, 14, 21, 28, ...

(viii) **E** The furniture mover's problem. Given a room with a door and a set of furniture, is it possible to move all the furniture into the room through that door?

(ix) **E,F,G** The set of all positions of Chinese GO, on a board of any size, from which white can win. (You will need to look this up.)

(x) **A** The Dyck language.

(xi) **D** The Jigsaw problem. (That is, given a finite set of two-dimensional pieces, can they be assembled into a rectangle, with no overlap and no spaces.)

(xii) **C** Factorization of binary numerals.

(xiii) **A** Boolean matrix multiplication.

(xiv) **I** All C++ programs which do not halt if given themselves as input.

(xv) **D** SAT.

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

(xvii) **A** 2-SAT.

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

(xix) **D** The Subset Sum Problem.

(xx) **D** The Block sorting problem.

(xxi) **E** The Sliding block problem.

(xxii) **D** The Hamiltonian cyle problem.

(xxiii) **D** The Traveling salesman problem.

(xxiv) **C** The Graph isomorphism problem.

(xxv) **D** The 3-coloring problem.

(xxvi) **A** The 2-coloring problem.

(xxvii) **L** TOT, the set of all machine descriptions $\langle M \rangle$ such that M halts on all possible inputs.

5. Fill in the ACTION table and GOTO table of an LALR parser or the following annotated context-free grammar. Your table should enforce the convention that both operations are left-associative, and that multiplication has higher precedence than subtraction, where x represents any identifier.
   1. $E \to E -_2 E_3$
   2. $E \to E *_4 E_5$
   3. $E \to x_6$

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

I will now try to explain how I was able to fill in the tables.

We know that E an be pushed onto the bottom, that is, onto stack state 0, and we also know that it must then have stack state 1. From the subscripts in the grammar, we know that E must be given stack state 3 when it is pushed onto stack state 2, and stack state 5 when it is pushed onto stack state 3. Thus, we have all entries of the GOTO table.

From the subscripts in the grammar, we know that when $-$, $*$, or x is pushed, it must be given stack state 2, 4, or 6, respectively. If the lookahead symbol is x we always shift if the stack is empty or the top symbol is an operator. If the lookahead symbol is an operator, we might shift or reduce. How can you tell? If shifting would violate a precedence rule, you reduce, otherwise you shift. Thus, you cannot shift $-$ if there is already an operator on the stack, and you cannot shift $*$ if there is already another $*$ on the stack. But you can shift $*$ if there is $-$ on the stack.

If the top stack state is 6, you always execute r3. Otherwise, you reduce if the top stack symbol is either 3 or 5 and shifting is not allowed, as described above, and if the lookahead symbol is either an operator of the end-of-file symbol, \$.

Practice the some input strings. At each step, ask what action will do the right thing. Here is my suggested list of practive inputs, each followed by the output generated by the parser. These should give you enough hints to fill in the tables.

$x\$$	3
$x - x\$$	331
$x * x\$$	332
$x - x - x\$$	33131
$x - x * x\$$	33321
$x * x - x\$$	33231
$x * x * x\$$	33322 **Error : it′sreally33232**

6. Consider the following annotated CF grammar G.
   1. $E \to E -_2 E_3$
   2. $E \to -_4 E_5$
   3. $E \to E \wedge_6 E_7$
   4. $E \to ({}_8 E_9)_{10}$
   5. $E \to x_{11}$

   Here is an LALR parser for G. Walk through the computation for the input string $x - -(-x \wedge x \wedge -x)$.

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

STACK	INPUT	OUTPUT	ACTION
$\$_0$	$x--(-x \wedge x \wedge -x)\$$		
$\$_0 x_{11}$	$--(-x \wedge x \wedge -x)\$$		$s11$
$\$_0 E_1$	$--(-x \wedge x \wedge -x)\$$	5	$r5$
$\$_0 E_1 -_2$	$-(-x \wedge x \wedge -x)\$$	5	$s2$
$\$_0 E_1 -_2 -_4$	$(-x \wedge x \wedge -x)\$$	5	$s4$
$\$_0 E_1 -_2 -_4 (_8$	$-x \wedge x \wedge -x)\$$	5	$s8$
$\$_0 E_1 -_2 -_4 (_8 -_4$	$x \wedge x \wedge -x)\$$	5	$s4$
$\$_0 E_1 -_2 -_4 (_8 -_4 x_{11}$	$\wedge x \wedge -x)\$$	5	$s11$
$\$_0 E_1 -_2 -_4 (_8 -_4 E_5$	$\wedge x \wedge -x)\$$	55	$r5$
$\$_0 E_1 -_2 -_4 (_8 E_9$	$\wedge x \wedge -x)\$$	552	$r2$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6$	$x \wedge -x)\$$	552	$s6$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 x_{11}$	$\wedge - x)\$$	552	$s11$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7$	$\wedge - x)\$$	5525	$r5$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7 \wedge_6$	$-x)\$$	5525	$s6$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7 \wedge_6 -_4$	$x)\$$	5525	$s4$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7 \wedge_6 -_4 x_{11}$	$)\$$	5525	$s11$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7 \wedge_6 -_4 E_5$	$)\$$	5525	$r5$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7 \wedge_6 E_7$	$)\$$	552552	$r2$
$\$_0 E_1 -_2 -_4 (_8 E_9 \wedge_6 E_7$	$)\$$	5525523	$r3$
$\$_0 E_1 -_2 -_4 (_8 E_9$	$)\$$	55255233	$r3$
$\$_0 E_1 -_2 -_4 (_8 E_9)_{10}$	$\$$	55255233	$s10$
$\$_0 E_1 -_2 -_4 E_5$	$\$$	5525542334	$r4$
$\$_0 E_1 -_2 E_3$	$\$$	5525523342	$r2$
$\$_0 E_1$	$\$$	55255233421	$r1$
$\$_0 E_1$	$\$$	55255233421	HALT

7. Let N be a large integer, and $w = \langle N \rangle$ its binary numeral. Let $n = |w|$. Explain how $O(n)$ parallel processors can decide, in $O(\log n)$ time, whether N is divisible by 3.

In general, given a base b numeral of length n, how can $O(n)$ parallel processesors decide, in $O(\log n)$ time, whether N is divisible by a given number m?

One method is to design a DFA, with m states, which accepts that set of numerals, where leading zeros are allowed. The start state is the only final state. Write an $m \times m$ Boolean matrix T_d for each digit d from 0 to $b-1$. For a given numeral $\langle N \rangle$, let d_i be its i^{th} digit, and let $M_i = T_{d_i}$. The product matrix $M_1 \times \cdots M_n$ can be computed in $O(\log n)$ time using n processors. and the $(0,0)^{\text{th}}$ entry of that product matrix is 1 if and only if N is divisible by m.

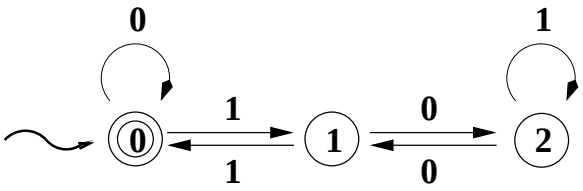
There are also two solutions which use the following properties of numerals.

(i) The value of a base b numeral is divisible by $b-1$ if and only if the sum of its digits is divisible by $b-1$

(ii) The value of a base b numeral is divisible by $b+1$ if and only if the alternating sum of its digits is divisible by $b+1$.

Example: in this case, N is a divisible by 3.
1011111111000011101010111100110011010100010011011111100010001011110000001011110001111110010111000010

We explain the DFA method first. The following DFA M accepts the language of all Boolean strings which are numerals for multiples of 3, and where leading zeros are allowed.



We use the method given in the handout regNC.pdf to decide whether the string is accepted by M.

Now consider the method given in (i) above. Since 4 is a power of 2, we can change a base 2 numeral to a base 4 numeral easily, by replacing each pair 00 by 0, 01 by 1, 10 by 2, and 11 by 3. For our example

1011111111000011101010111100110011010100010011011111100010001011110000001011110001111110010111000010
we obtain:
2 3 3 3 3 0 0 3 2 2 2 3 3 0 3 0 3 1 1 0 1 0 3 1 3 3 2 0 2 0 2 3 3 0 0 0 2 3 3 0 1 3 3 2 1 1 3 0 0 2
We then use the tournament method to find the sum of those numerals in $O(\log n)$ time. The N is a multiple of 3 if and only if that sum is a multiple of 3.

We now consider the method of (ii). We replace each pair of digits by their alternating sum. Thus we replace 11 by 0, 01 by 1, 10 by -1, and 00 by 0. For our example, we obtain

-1 0 0 0 0 0 0 0 -1 -1 -1 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 -1 0 -1 0 -1 0 0 0 0 0 -1 0 0 0 1 0 0 -1 1 1 0 0 0 -1
We then use the tournament method to find the sum of those values in $O(\log n)$ time. That sum is a multiple of 3 if and only if N is a multiple of 3.

8. State the pumping lemma for regular languages. Do not get confused this time!

   For any regular language L, there exists an integer p, called the pumping length of L, such that for any $w \in L$ of length at least p, there exist strings x, y and z such that the following four statements hold:

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

9. Prove that $L = \{a^n b^n : n \geq 0\}$ is not regular.

   By contradiction. Assume L is regular. Let p be the pumping length of L. Let $w = a^p b^p \in L$. Since $|w| = 2p > p$ we can pick strings x, y, z, such that:

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

   xy is a prefix of w of length no greater than p, and hence is a substring of a^p, thus $y = a^k$ for some $k > 0$. Let $i = 0$. Then Then $xz = a^{p-k}b^p \notin L$ since $p - k < p$. But by 4., $xz \in L$, contradiction. We conclude that L is not regular.

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

    We construct a $\mathcal{P}$–TIME function R. Let $E = C_1 * C_2 * \cdots * C_K$ be an instance of 3-SAT. Let each $C_i = t_{i,1} + t_{i,2} + t_{i,3}$, where $t_{i,j}$ is either a variable or the negation of a variable.

    We construct $R(E) = (G, K)$, an instance of IND, where the vertices of G are $v_{i,j}$ for $1 \leq i \leq K$ and $1 \leq j \leq 3$. G has an edge from $v_{i,j}$ to $v_{i',j'}$ if and only if either $i = i'$ or $t_{i,j} * t_{i',j'}$ is a contradiction. Then G has an independent set of K vertices if and only if E is satisfiable.

11. Prove that the halting problem is undecidable.

    By contradiction. Assume that the halting problem is decidable, that is, there is a machine H such that $H(< M >, w) = 1$ if and only if the machine M halts with input w. Let Q be a machine which is equivalent to the following program:

    Read a machine desciption $\langle M \rangle$.
    If $H(\langle M \rangle, \langle M \rangle) = 1$, run forever.
    Else halt.

    Now, run Q with input $\langle Q \rangle$. If $H(\langle Q \rangle, \langle Q \rangle) = 1$, then Q runs forever, contradiction, while if $H(\langle Q \rangle, \langle Q \rangle) = 0$, Q halts, contradiction. We conclude that H cannot exist, hence the halting problem is undecidable.