1. True/False/Open
   (i) F Every subset of a regular language is decidable.
   (ii) T The intersection of any two \( \mathcal{N} \mathcal{P} \) languages is \( \mathcal{N} \mathcal{P} \).
   (iii) T Every language accepted by a non-deterministic machine is accepted by some deterministic machine.
   (iv) O \( \mathcal{N} \mathcal{C} = \mathcal{P} \).
   (v) O \( \mathcal{P} = \mathcal{N} \mathcal{P} \).
   (vi) O The Boolean Circuit Problem (CVP) is in \( \mathcal{N} \mathcal{C} \).
   (vii) O The independent set problem is \( \mathcal{P} \)-time.
   (viii) T IF \( L_1 \) is undecidable and there is a recursive reduction of \( L_1 \) to \( L_2 \), then \( L_2 \) must be undecidable.
   (ix) T If \( S \) is a recursive set of positive integers, then \( \sum_{n \in S} 2^{-n} \) must be a recursive real number.
   (x) T Multiplication of matrices with binary numeral entries is \( \mathcal{N} \mathcal{C} \).
   (xi) T Equivalence of regular expressions is decidable.
   (xii) T Every recursively enumerable language is generated by a general grammar.
   (xiii) T Equivalence of context-free grammars is co-\( \mathcal{R} \mathcal{E} \).
   (xiv) T The language consisting of all fractions whose values are less than the natural logarithm of 5.0 is recursive.
(xv) **T** If \( L \) is in \( \mathcal{RE} \) and also co-\( \mathcal{RE} \), then \( L \) must be decidable.

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

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

(xviii) **T** Every sliding block problem is \( \mathcal{P} \)-space.

(xix) There are uncountably many co-\( \mathcal{RE} \) languages.

This problem is incorrectly stated. An alphabet must be specified, in which case the answer is \( \text{F} \).

(xx) **T** If \( L \) is any \( \mathcal{P} \)-time language, there is an \( \mathcal{NC} \) reduction of \( L \) to CVP, the Boolean circuit problem.

(xxi) **T** There is a polynomial time algorithm for checking whether an integer is prime.

This was open until recently.

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

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

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

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

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

### 2.

Every language, or problem, falls into exactly one of these categories. For each of the languages, write a letter indicating the correct category. [5 points each]

- **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 decidable, but not known to be \( \mathcal{P} \)-space.
- **G** \( \mathcal{RE} \) but not decidable.
- **H** co-\( \mathcal{RE} \) but not decidable.
- **I** Neither \( \mathcal{RE} \) nor co-\( \mathcal{RE} \).

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

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

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

(d) **H** \( \{ \langle G \rangle : L(G) \text{ is the Dyck language} \} \)
(e) **E** All positions of RUSH HOUR from which it is possible to win.

(f) **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.)

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

3. [20 points] Find a DFA equivalent to the NFA shown in Figure 1.

![Figure 1: NFA for problems 3 and 4](image)

Figure 1: NFA for problems 3 and 4

![Figure 2: Equivalent DFA](image)

Figure 2: Equivalent DFA

4. [20 points] Give a regular grammar for the language accepted by the machine in Figure 1.

\[
S \rightarrow aS \\
S \rightarrow bS \\
S \rightarrow aB \\
S \rightarrow cB \\
B \rightarrow aC \\
B \rightarrow bC \\
B \rightarrow cC \\
C \rightarrow bC \\
C \rightarrow \lambda
\]

5. [20 points] Give a regular expression for the language accepted by the machine in Figure 3

\[a^*b(ca^*b + a + ac^*b)^*\]
Figure 3: NFA for problem 5.

6. Which class of languages does each of these machine classes accept? [5 points each]
   (a) Deterministic finite automata. **Regular Languages**
   (b) Non-deterministic finite automata. **Regular Languages**
   (c) Push-down automata. **Context-Free Languages**
   (d) Turing Machines. **Recursively Enumerable Languages**

7. [20 points] Let $L = \{ w \in \{a,b\}^* : \#_a(w) = \#_b(w) \}$, that is, each string of $L$ has equal numbers of each symbol. Draw a PDA which accepts $L$.

8. [20 points] 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.)

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$-$</th>
<th>$*$</th>
<th>$($</th>
<th>$)$</th>
<th>$$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$0$</td>
<td>$s11$</td>
<td>$s4$</td>
<td>$s8$</td>
<td>$S$</td>
<td>$1$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$1$</td>
<td>$s2$</td>
<td>$s6$</td>
<td>$\text{halt}$</td>
<td>$3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$2$</td>
<td>$s11$</td>
<td>$s4$</td>
<td>$s8$</td>
<td>$s1$</td>
<td>$1$</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$3$</td>
<td>$r1$</td>
<td>$s6$</td>
<td>$r1$</td>
<td>$r1$</td>
<td>$5$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$4$</td>
<td>$s11$</td>
<td>$s4$</td>
<td>$s8$</td>
<td>$r2$</td>
<td>$r2$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$5$</td>
<td>$r2$</td>
<td>$r2$</td>
<td>$r2$</td>
<td>$r2$</td>
<td>$7$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$6$</td>
<td>$s11$</td>
<td>$s4$</td>
<td>$s8$</td>
<td>$r3$</td>
<td>$r3$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$7$</td>
<td>$r3$</td>
<td>$r3$</td>
<td>$r3$</td>
<td>$r3$</td>
<td>$9$</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>$8$</td>
<td>$s11$</td>
<td>$s4$</td>
<td>$s8$</td>
<td>$s2$</td>
<td>$s6$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>$9$</td>
<td>$r4$</td>
<td>$r4$</td>
<td>$r4$</td>
<td>$r4$</td>
<td>$r4$</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>$10$</td>
<td>$r5$</td>
<td>$r5$</td>
<td>$r5$</td>
<td>$r5$</td>
<td>$r5$</td>
<td></td>
</tr>
</tbody>
</table>
9. [20 points] Prove that any decidable language can be enumerated in canonical order by some machine.

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

For all $i$ from 1 to $\infty$

If $w_i \in L$

Write $w_i$

10. [20 points] Give a polynomial time reduction of 3-SAT to the independent set problem.

Let $E = C_1 \ast C_2 \ast \cdots \ast C_k$ be a Boolean expression in 3-CNF form. Let $C_i = t_{i,1} + t_{i,2} + t_{i,3}$ where each $t_{i,j}$ is either a variable or the negation of a variable. Let $G$ be a graph of $3k$ vertices $V = \{v_{i,p} : 1 \leq i \leq k, 1 \leq p \leq 3\}$

There is an edge between $v_{i,p}$ and $v_{j,q}$ if and only if either $i = j$ or $t_{i,p} \ast t_{j,q}$ is a contradiction.

11. [20 points] Prove that the halting problem is undecidable.

Prove that the halting problem is undecidable.

By contradiction. Suppose the halting problem is decidable. Let $L_{\text{diag}}$ be the set of programs which do not halt if given themselves as input. The following program, which we call $P_{\text{diag}}$, accepts $L_{\text{diag}}$:

read a program $P$

if $P$ halts with input $P$

Enter an infinite loop

else HALT

Note that the condition in the second line of $P_{\text{diag}}$ can always be evaluated because the halting problem is decidable.

We now ask, is $P_{\text{diag}}$ a member of $L_{\text{diag}}$? The answer cannot be either yes or no. Thus, the halting problem is undecidable.