1. (5 pts.) Construct a NO instance of 3SAT over 3 variables \( U = \{x, y, z\} \). That is, construct a valid CNF expression over 3 variables, each of whose clauses contains exactly 3 literals, that is not satisfiable.

**ANSWER:** All 8 combinations of \( x, y, z \) and complements. For any assignment to \( U \), exactly one of the 8 clauses will have all its literals assigned false, while the other 7 clauses are satisfied.

2. (7 pts.) Let \( M_{SAT} \) be a nondeterministic polynomial time bounded TM for SAT. Assume that on all inputs, every computational path of \( M_{SAT} \) terminates in either \( q_{accept} \) or \( q_{reject} \). Define the complement machine \( M^c \) as usual, i.e., formed by switching the \( q_{accept} \) and \( q_{reject} \) states. Describe the language of the machine \( M^c \).

**ANSWER:** The complement of \( \text{TUTOLOGY} \). That is, \( M^c \) accepts \( X \) iff \( X \) is not a tautology. (Assuming non-instances are rejected by both machines; otherwise, \( M^c \) also accepts all non-instances.)

3. (10 pts.) We proved the zero-one pair-function minimization problem (0/1-PFM) to be NP-complete in HW 4. Consider a polynomial time mapping reduction \( 2\text{SAT} \xmapsto{\alpha_m} 0/1\text{-PFM} \) from 2SAT to the 0/1-PFM in which each 2SAT variable \( u_i \) maps to 0/1-PFM variable \( x_i \), and each 2SAT clause \( c_j = \{z_i, z_k\} \) maps to a 0/1-PFM function \( f_j(x_i, x_k) \) which assumes the value 1 if both variables are assigned counter to the sign of their corresponding literals in \( c_j \) and assumes the value 0 otherwise. For example, if \( c_j = \{u_2, \overline{u}_3\} \) then \( f_j(x_2, x_3) \) evaluates as \( f_j(0, 1) = 1 \) and \( f_j(\cdot, \cdot) = 0 \) on the remaining 3 possible value combinations of \( x_2 \) and \( x_3 \).

a) What should be the bound \( B \) of the constructed 0/1-PFM instance in order for the above to define a valid polynomial time mapping reduction from 2SAT to 0/1-PFM?

b) Since it is obvious that 2SAT is in NP, why does this reduction fail to establish that 2SAT is NP-complete?

**ANSWERS**

a) \( B = 0 \). Clause \( c_j \) not satisfied by an assignment iff \( f_j = 1 \)

b) Reducing \( 2\text{SAT} \xmapsto{\alpha_m} 01\text{-PFM} \) is the wrong direction for this conclusion. Of course, all problems in P poly reduce to any problem. map_f() solves the problem and maps to yes/no appropriately.

4. (20 pts.) Let \( k > 0 \) be a fixed integer constant. Define an \( FA_k \) over finite input alphabet \( \Sigma \) to be a deterministic finite automaton with the addition of a two-way read/write auxiliary work tape consisting of exactly \( k \) squares. Each of these squares contains a symbol from the finite tape alphabet \( \Gamma \). Similar to a PDA and 2-tape TM model, the transition function \( \delta \) of an \( FA_k \) maps from (state \( q \), current input symbol \( c \in \Sigma \), current work-tape symbol \( t \in \Gamma \) ) to
(state \(q', \) new work-tape symbol \(t' \in \Gamma, \) work-tape head direction \(L \) or \(R\)).
The input tape remains 1-way, read-only (like that of an FA and PDA), and the work-tape
head cannot move to the left of square \(1\) or to the right of square \(k\) --- an attempt to do so
leaves the head in its previous position. Consider, for a fixed integer constant \(k > 0,\)
the class of languages recognized by the \(FA_k\) model. Where does this class of languages fit
on the language hierarchy? How does the answer depend on what fixed value \(k > 0\) you
consider? Justify your answer.

ANSWER Since \(k\) is fixed, max number of configs of work tape is fixed at \(k^{|\Gamma|^k},\)
independent of length of input (unlike an LBA, where this depends on input length). FA
M simulates \(FA_k\) Mk by crossing each of Mk’s states with a set of states representing
the \(k^{|\Gamma|^k}\) possible work-tape configurations. Transitions then can be performed as:
\((q \times \text{config}, c)\rightarrow(q' \times \text{config}')\) where config' represents the work tape
configuration that follows from config when Mk makes its transition from state \(q\)
scanning given input symbol \(c\) and the tape symbol specified by config. Hence, the class
of languages recognized by the \(FA_k\) model, for any \(k,\) is the class of regular languages.

5. (30 pts. total) Consider the following language over \(\Sigma^*\).
\(L = \{ <M> \mid M \text{ is a TM and } L(M) \text{ contains at least one string from } \Sigma^* \} \).
a) (5 pts.) Demonstrate that \(L\) is recursively enumerable.
b) (15 pts.) Construct a mapping reduction from \(A_{TM}\) to \(L,\) i.e. \((A_{TM} \leq_m L)\).
What does the existence of this reduction allow you to conclude about \(L?\)
c) (5 pts.) Describe a Turing reduction (i.e., an oracle reduction) from \(EMPTY_{TM}\) to \(L,\)
thus demonstrating that \(EMPTY_{TM}\) is decidable relative to \(L.\)
d) (5 pts.) What can you conclude about the existence of a mapping reduction from
\(EMPTY_{TM}\) to \(L?\) Explain.

ANSWER
a) Demonstrate that \(L\) is RE.
Dovetail through \(\sigma^*\), calling \(M,\) halting in q_accept if \(M\) ever accepts an \(x_i.\)
b) Construct a mapping reduction from \(A_{TM}\) to \(L,\) i.e. \((A_{TM} \leq_m L)\)
What does the existence of this reduction allow you to conclude about \(L?\)
\(<M, w>, \) instance of \(A_{TM},\) maps to \(<M_1(w)>,\) where TM \(M_1(w)\) is described as follows:
\(M_1(w),\) on input \(x,\) compares \(x\) to \(w.\) If \(x \neq w, M_1(w)\) rejects
\(x;\) otherwise, \(M_1(w)\) simulates \(M\) on \(w\)
Note that \(M\) accepts \(w\) iff \(M_1(w)\) accepts \(w.\) Further, \(M_1(w)\) can accept no string other
than \(w.\) Hence \(<M, w> \) in \(A_{TM}\) iff \(<M_1(w) > \) in \(L.\) The existence of this reduction
establishes that \(L\) is not decidable.
c) Describe an Turing reduction (i.e., an oracle reduction) from
\(EMPTY_{TM}\) to \(L\) that shows \(EMPTY_{TM}\) is decidable relative to \(L.\)
A decider for \(EMPTY_{TM}\) utilizes an oracle \(M_L\) for \(L\) by calling \(M_L\) once on its input
\(<M>\) and returning the opposite answer. Note, of course, \(L\) and \(EMPTY_{TM}\) are complements.
d) What can you conclude about the existence of a mapping reduction from
EMPTY_TM to L? Explain. It cannot exist. Since L is undecidable and RE, its complement EMPTY_TM cannot be RE. But if we had the mapping reduction from EMPTY_TM to L, since L is RE, this would imply EMPTY_TM is also RE.

6. (28 pts. total) Modify the 3SAT to Vertex Cover (VC) reduction $3SAT \rightarrow VC$ so that there are no edges between any pairs of the $a_p[j]$ vertices (what Sipser refers to as the vertices in the clause gadget structures). In particular, in this modified reduction (which will not result in a valid reduction), the vertices are the same as in the original reduction, but the only edges are edge $\{u_i, \bar{u}_i\}$, between the positive and negative versions of each 3SAT variable $u_i$ ($i = 1, 2, \cdots, n$), and edge $\{z_i, a_p[j]\}$, exactly when $z_i$ (which is either $u_i$ or $\bar{u}_i$) is the $p^{th}$ ($p = 1, 2, \text{or} 3$) literal of clause $c_j$ ($j = 1, 2, \cdots, m$). The bound remains at $K = (2m + n)$, as before.

a) (5pts.) Under the modified reduction, every YES instance of 3SAT still maps to a YES instance of VC. Argue why this is the case.

b) (8pts.) Under the modified reduction, some NO instances of 3SAT will map to YES instances of VC, and hence the modified reduction is not a valid reduction. Supply the intuition as to why a NO instance of 3SAT can map to a YES instance of VC under the modified reduction.

c) (15pts.) More formally, establish the claim of part (b) by specifying a specific NO instance of 3SAT (i.e., an unsatisfiable CNF expression, with exactly three literals per clause) that the modified reduction maps to a YES instance of VC. Demonstrate that your mapped to instance of VC is indeed a YES instance of VC by identifying a vertex cover of the required size.

**ANSWER**

a) Since the only modification is removing edges, any $V'$ subset $V$ that is a cover under the old mapping is a cover under the new mapping.

b) One problem is, one clause might have $> 1$ true literal and another none, i.e., it allows "sharing" of the coverage. A clause $c_j$ with more than 1 true literal requires fewer than 2 of its $ap[j]$ vertices included in a cover, allowing a clause $c_j'$ to contribute to the cover all three of its $ap[j']$ to make up for the fact that it has no true literal. Note how the inclusion of the clause edges prevents this opportunity, because no matter what the structure of a clause, 2 of the 3 $ap[j]$ must be selected in a cover, and hence no "sharing" of excess is possible.

c) Take the 8 clause example from Problem 1. Consider the assignment TTT. Only the last clause $\{x', y', z'\}$ fails to be satisfied. Select (positive) $x, y, z$ for inclusion in the cover, and 2 $ap[j]$ so as to cover any uncovered edges in each of the remaining 7 clauses. Select 2 arbitrary $ap[8]$ for $c_8$. This set of vertices covers every edge, except for one $\{ap[8], L'\}$ edge from clause 8. But (for example) clause $c_1$ has vertices to spare. Since $c_1$ connects to $x, y, z$, none of the $ap[1]$ vertices from this clause is needed in the cover. Swap out one of these and include instead the unselected $ap[8]$ in $c_8$ and we have a cover of the required size.

An alternative example can be formed for any NO instance of 3SAT in which the number $n$ of variables is no more than twice the number $m$ of clauses. In this case, $2n \leq 2m + n$, and a vertex cover of the required size can be formed simply by selecting every (positive and negative) literal vertex.