Solutions to some sample 431 final exam questions

1. Consider the following list of properties that might apply to the stated language.

T-rec: The language is Turing-recognizable.
Dec: The language is decidable.
NP: The language is in NP.
NP-c: The language is NP-complete.
$\mathcal{P}$: The language is in $\mathcal{P}$.

Circle all the properties that you are certain are true.
× out all the properties that you are certain are false.
NOTE: You may not be able to do either for some properties.

(a) $\{\langle M, w\rangle \mid$ Turing machine M accepts $w\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(b) $\{\langle M, w\rangle \mid$ Turing machine M accepts w in at most $\lvert w\rvert$ steps$\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(c) $\{\langle M, w\rangle \mid$ Turing machine M accepts w in at most $2^{\lvert w\rvert}$ steps$\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(d) $\{\langle M, w\rangle \mid$ Turing machine M does not accept $w\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(e) $L(\alpha)$ for some regular expression α.	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(f) $\{\langle F\rangle \mid F$ is a 3-CNF formula which evaluates to true on some truth assignment$\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(g) $\{\langle F, x\rangle \mid F$ is a 3-CNF formula which evaluates to true on truth assignment $x\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(h) $\{\langle F\rangle \mid F$ is a propositional logic tautology$\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$
(i) $\{\langle G, H\rangle \mid G$ and H are isomorphic graphs$\}$	T-rec	Dec	NP-c	NP	$\mathcal{P}$

2. We prove that $A_{TM} \leq_m L_2$ as follows:

The mapping reduction f takes input $\langle M, w\rangle$ and produces output $\langle M_w\rangle$ where M_w is the Turing machine defined in class but is now over the input domain $\{0,1,2\}^*$.

That is M_w on input x, ignores x and runs M on input w and does what M would do.

By construction $L(M_w) = \begin{cases} \{0,1,2\}^* & \text{if } M \text{ accepts w} \\ \varnothing & \text{if not.} \end{cases}$

There will be a 2 in $L(M_w)$ iff M accepts w iff $\langle M, w\rangle$.

4. (b) This algorithm is a specific algorithm that does not capture all algorithms that could be tried to solve the problem. For example, the same algorithm as given could be used to solve matrix equations modulo 2 and would take exponential time to do so; however, in that case there is an alternative algorithm, Gaussian elimination, that can be used to solve the matrix equations problem in polynomial time.

5. We show that *SET-PARTITION* is NP-complete.

1. *SET-PARTITION* $\in$ NP:
   (a) Guess a binary string of length n representing a subset $S \subseteq \{1, \ldots, n\}$.

(b) Verify that $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$.

(c) This polynomial time to check since we can compute the two sums in polynomial times.

2. We show that $SET\text{-}PARTITION$ is NP-hard by showing that $SUBSET\text{-}SUM \leq_m^p SET\text{-}PARTITION$.

(a) On input $\langle x_1, \ldots, x_m, t \rangle$ for $SUBSET\text{-}SUM$, define $M = \sum_{i=1}^{n} x_i$. Assume that $t \leq M$ – if not we simply map the input to $\{1, 2\}$. Otherwise, using the hint, remove t, let $n = m + 2$ and add two extra numbers $x_{n-1} = M + t$ and $x_n = 2M - t$.

(b) The computation is clearly polynomial time since it simply requires the computation of M and the two extra numbers.

(c) Correctness ($\Rightarrow$): Suppose that $\langle x_1, \ldots, x_m, t \rangle \in SUBSET\text{-}SUM$. Then there is a subset $S' \in \{1, \ldots, m\}$ such that $\sum_{i \in S'} x_i = t$ and $t \leq M$. Therefore the output has $n = m + 2$ values and we define $S \subseteq \{1, \ldots, n\}$ by $S = S' \cup \{n\}$. Then $\sum_{i \in S} x_i = t + 2M - t = 2M$ and $\sum_{i \notin S} x_i = M + t + \sum_{i \leq m, i \notin S'} = M + t + M - t = 2M$ and hence $\langle x_1, \ldots, x_n \rangle \in SET\text{-}PARTITION$ as required.

(d) Correctness ($\Leftarrow$): Suppose that $\langle x_1, \ldots, x_n \rangle \in SET\text{-}PARTITION$ Then we know that we have a subset $S \in \{1, \ldots, n\}$ such that $\sum_{i \in S} x_i = \sum_{i \notin S} x_i$. By definition of the reduction we also know that the sum of all the elements is $4M$ so the sum of each side is $2M$. Because x_{n-1} and x_n add up to $3M$, which is too large, we can't have both elements in S or both elements not in S. Therefore, one of S or $\overline{S}$ contains n but not $n - 1$. Assume, without loss of generality that S does. (If not, simply complement S.) Then define $S'' = S - \{n\}$ and observe that $S'' \subseteq \{1, \ldots, m\}$. Then $2M = \sum_{i \in S} x_i = 2M - t + \sum_{i \in S''} x_i$. It follows that $\sum_{i \in S''} x_i = t$ and hence $\langle x_1, \ldots, x_m, t \rangle \in SUBSET\text{-}SUM$ as required.

6. We show that TSP is NP-complete.

1. $TSP \in$ NP:

(a) Guess a length n sequence of integers $i_1, \ldots i_n$, each in the range 1 to n.

(b) Verify that all the numbers are different and that

$$c_{i_1 i_2} + \cdots c_{i_{n-1} i_n} + c_{i_n i_1} \leq K.$$

(c) This polynomial-time to check.

2. We show that TSP is NP-hard by showing that $DHAMCYCLE \leq_m^p TSP$.

(a) For the reduction, on input an undirected graph $G = (V, E)$, number the vertices of G as $v_1, \ldots, v_n$, define the cost matrix C by

$$c_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 2 & \text{if } (v_i, v_j) \notin E \end{cases}$$

and set $K = n$.

(b) The computation is clearly polynomial time.

(c) Correctness ($\Rightarrow$): Suppose that G has a directed Hamiltonian cycle. Then the list of those indices in the order along the cycle starting at any spot will yield a TSP tour of cost n. Therefore all of the n edges along the cycle will have cost 1 in the TSP instance so their total cost will be n.

(d) Correctness ($\Leftarrow$): Suppose that (c, K) is a yes instance for TSP. Since there are n distances c_{ij} contributing to the sum and each distance in C is at least 1, if they are to sum to at most $K = n$, then every such edge counted must have $c_{ij} = 1$. By definition, this means that $(v_i, v_j) \in E$ for all n edges. These edges form a directed Hamiltonian cycle in G.

7. Consider the following algorithm M':

On input $\langle M, a, b \rangle$
    For each string $x \in \Sigma^*$ in lexicographic order do
        Simulate M on input x for $a|x|^2 + b$ steps
            If M has not yet halted then accept.
    End For

If there is some string x where M runs for more than $a \cdot |x|^2 + b$ steps then this algorithm will accept. Otherwise, M' will run forever. Therefore $L(M') = L$ and hence L is Turing-recognizable.