Hardness is given as [x]. Higher number means harder. Marks as (x). Assume the input alphabet is $\{0, 1\}$ unless mentioned otherwise.

1. Define a TM M such that for any string x, we have $q_0 x \vdash^* q_h \overline{x}$, where $\overline{x}$ is the string where all bits in x are flipped (0s changed to 1 and 1s to 0) and q_h is a halting state. [1] (1)
2. For question 1, show the sequence of configurations for input 1001. [1] (1)
3. Suppose the input and tape alphabets of a TM are $\Sigma = \{1\}$ and $\Gamma = \Sigma \cup \{\sqcup\}$. Show that if a language A is decided in poly-time by such a TM, then A is decidable in logspace by another (possibly with a larger tape alphabet) TM. [2] (1)
4. Argue that it is unlikely that we could prove: For all languages A if $A \leq_p$ 3SAT, then $A \in$ NP. [2] (1)
5. Define $\leq^m_{\log}$ reductions. [1] (1)
6. Show that $A \leq^m_{\log} B$ and $B \in$ LOGSPACE implies $A \in$ LOGSPACE. [2] (1)
7. Suppose the input to 3SAT problem is guaranteed to have each variable appear at most once in the given formula. The appearance could be as x or its negation $\overline{x}$. Show that this problem has a poly-time algorithm. [1] (1)
8. Suppose the input to 3SAT problem is guaranteed to have each variable appear at most twice in the given formula. One appearance could be as x and the other as $\overline{x}$. Show that this problem has a poly-time algorithm. [2] (1)
9. Suppose the input to 3SAT problem is guaranteed to have each variable appear at most thrice in the given formula. Show that this problem is $\leq^m_p$-complete for NP. [3] (1)
10. State the verifier and non-deterministic TM based definitions for the class NP and argue that they are equivalent. [1] (1)
11. For any language A, we can define $A^* = \cup_{i \geq 0} A^i$ as usual. Show that if $A \in$ NP, then $A^* \in$ NP. [3] (1)
12. Show that if $A \in$ P, then $A^* \in$ P. [3] (1)
13. The language Σ^* is not complete for P with $\leq^m_p$ reductions but it is complete for P with $\leq_p$ reductions. Explain. [2] (1)
14. A homomorphism from Σ to Σ' is a map $\phi : \Sigma^* \mapsto \Sigma'^*$ such that we define $\phi(a)$ for all $a \in \Sigma$ and extend it to all strings such that $\phi(xy) = \phi(x)\phi(y)$. Given a homomorphism ϕ and a language $A \in \Sigma^*$, we define the language $\phi(A) = \{y \in \Sigma'^* \mid y = \phi(x) \text{ for some } x \in A\}$. Show that P = NP iff P is closed under ϵ-free homomorphisms (homomorphisms where $\phi(a) \neq \epsilon$ for all $a \in \Sigma$). That is, P = NP iff for any language A and any ϵ-free homomorphism ϕ, $A \in$ P implies $\phi(A) \in$ P. [3] (1)
15. Show that the language ADD $= \{a\#b\#c \mid a + b = c\} \in$ LOGSPACE where a, b, c are binary numbers [3] (1)

Student's Scores
(To be filled by the instructor)

1	2	5	7	10	[1]
/1	/1	/1	/1	/1	/5
3	4	6	8	13	[2]
/1	/1	/1	/1	/1	/5
9	11	12	14	15	[3]
/1	/1	/1	/1	/1	/5

Instructor's Remarks
(To be filled by the instructor)

Hints

I omit those that directly follow from our discussions.

3. The logspace TM can keep the number of 1s in the tape of M and the input head index. Since M is poly-time, both numbers can be encoded using logarithmically many bits.
4. Using poly-time Turing reductions, we can reduce $A = \overline{\text{3SAT}}$ to 3SAT. If $A \in$ NP, we will have coNP = NP, which we believe is unlikely.
8. We can replace $(A \vee x) \wedge (B \vee \overline{x})$ by $A \vee B$. Do this until there are no variables that occurs positively and negatively.
9. Suppose x occurs $k > 3$ times in the formula. Replace the i^{th} occurence with a new variable x_i. Finally, we add the following clauses $(\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \cdots \wedge (\overline{x_{k-1}} \vee x_k) \wedge (\overline{x_k} \vee x_1)$ to the resulting formula. This enforces that all x_i must have the same value. Observe that each x_i occurs exactly three times.
11. To verify that $x \in A^*$, we require as certificate the strings $y_1, \ldots, y_i$ such that $x = y_1 \ldots y_i$ and for each y_j the certificate z_j that shows that $y_j \in A$.
12. We cannot try all possible ways to split the input string x as that would take exponential-time. There is a dynamic programming algorithm. Find it.
13. Take a language $A \in$ P and a string $x \notin A$, then for $A \leq_p^m \Sigma^*$, we need x to reduce to some string not in Σ^*, which does not exist. A poly-time Turing reduction can simply ignore Σ^* and directly run A's poly-time algorithm.
14. (only if) Suppose P = NP, $A \in$ P, and ϕ is an ϵ-free homomorphism. We show $\phi(A) \in$ NP. The certificate for a string $y \in \phi(A)$ is the $x \in A$ such that $y = \phi(x)$. Note that $|x| \leq |y|$. Since ϕ has a constant-size description (we need only images of symbols to describe it), we can compute it in linear-time.
    (if) Suppose P is closed under ϵ-free homomorphisms. To show P = NP, we just have to show that there is a language $A \in$ P and an ϵ-free homomorphism ϕ such that CIRCUITSAT $= \phi(A)$. Consider the language CIRCUITEVAL of strings (C, x) where $C(x) = 1$ as A. Now define a suitable encoding of this language and a homomorphism. The idea is to "erase" all information about x from this language.
15. We verify the bits of c one-by-one. Say c is m bits. We keep an index i that runs from 0 to $m - 1$. To verify the i^{th} bit of c, we first read the i^{th} bits of a and b (set to 0 if there is no such bit), and then check whether the output bit matches it. We also need to keep track of the carry bit (this can be done in the finite state of the TM). We start with $i = 0$ and keep incrementing i until the i^{th} symbol from the end becomes #.