In all problems the languages are over the alphabet $\Sigma = \{0, 1\}$.

**Problem 1**  [40 points] If $\phi$ is a boolean CNF formula then $\|\phi\|$ denotes the number of variables in it. If $a \in \{0, 1\}^{\|\phi\|}$ is an assignment to the variables of $\phi$ then we let $\text{NumSatClauses}(\phi, a)$ denote the number of clauses of $\phi$ that are satisfied by $a$. We then define the function $\text{MaxSAT}: \Sigma^* \to \mathbb{N}$ as follows: for any boolean formula $\phi$ we let

$$\text{MaxSAT}(\langle \phi \rangle) = \max_a \{ \text{NumSatClauses}(\phi, a) \}.$$  

Above, the maximum is over all assignments $a \in \{0, 1\}^{\|\phi\|}$ to the variables of $\phi$. (We also define $\text{MaxSAT}(w)$ to be 0 for any $w$ that does not encode a boolean CNF formula.) Prove the following:

1.  [10 points] If the function $\text{MaxSAT}$ is polynomial-time computable then $P = NP$.

   Assume $\text{MaxSAT}(\cdot)$ is polynomial-time computable. This means there is a polynomial-time TM $M$ that on input the encoding $\langle \phi \rangle$ of a boolean formula $\phi$, halts with output $\text{MaxSAT}(\langle \phi \rangle)$. We have to show that $P = NP$. We will do this by showing that SAT is in $P$. Since SAT is NP-complete, it follows that $P = NP$.

   To show that SAT is in $P$ we present a polynomial time decision algorithm $M'$ for it. It takes as input the encoding $\langle \phi \rangle$ of a CNF formula $\phi$. It uses algorithm $M$ as a subroutine:

   Algorithm $M'(\langle \phi \rangle)$
   
   Let $m$ be the number of clauses in $\phi$
   
   Let $k \leftarrow M(\langle \phi \rangle)$
   
   If $k = m$ then accept else reject

2.  [30 points] If $P = NP$ then the function $\text{MaxSAT}$ is polynomial-time computable.

   Assume $P = NP$. We want to show that $\text{MaxSAT}(\cdot)$ is polynomial-time computable. We begin by considering the language

   $$\text{CSAT} = \{ \langle \phi, k \rangle : \phi \text{ is a boolean formula and } k \in \mathbb{N} \text{ and } \text{MaxSAT}(\phi) \geq k. \}.$$  

   It is easy to see that CSAT is in $NP$. (A witness for membership of $\langle \phi, k \rangle$ in CSAT is an assignment to the variables of $\phi$ that satisfies at least $k$ clauses of $\phi$.) Now, our assumption $P = NP$ means there is a polynomial-time TM $M'$ that decides CSAT. Then the following TM computes $\text{MaxSAT}(\cdot)$:
Algorithm $M(w)$

If $w$ is not the encoding of a boolean formula then return 0
Let $\varphi$ be the boolean formula such that $w = \langle \varphi \rangle$
Let $m$ be the number of clauses in $\varphi$
For $k = m$ downto 1 do
  If $M'(\langle \varphi, k \rangle)$ accepts then return $k$
End For

This TM runs in polynomial time because $M'$ runs in polynomial time and the length of the input $\langle \varphi, k \rangle$ to $M'$ is bounded by a polynomial in the length of the input $\langle \varphi \rangle$ to $M$.

---

Problem 2 [30 points] Let $\text{NP}^* = \text{NP} - \{\emptyset, \Sigma^*\}$ and let $	ext{NPC}$ denote the class of NP-complete languages. Prove the following:

1. [10 points] If $\text{NP}^* \subseteq \text{NPC}$ then $\text{P} = \text{NP}$

Assume $\text{NP}^* \subseteq \text{NP}$. We want to show $\text{P} = \text{NP}$. Since $\text{P} \subseteq \text{NP}$ we only need to show that $\text{NP} \subseteq \text{P}$. So let $A$ be an arbitrary language in $\text{NP}$. We want to show $A \in \text{P}$. Let $B$ be a language in $\text{P} - \{\emptyset, \Sigma^*\}$. (Many such languages exist). Since $\text{P} \subseteq \text{NP}$, $B$ is in $\text{NP}^*$, hence, by our assumption, is $\text{NP}$-complete. So $A \leq_p B$. Recall a basic property of polynomial time reductions: If $A \leq_p B$ and $B \in \text{P}$ then $A \in \text{P}$. But $B$ is indeed in $\text{P}$ so this implies $A$ is also in $\text{P}$, as desired.

2. [20 points] If $\text{P} = \text{NP}$ then $\text{NP}^* \subseteq \text{NPC}$

Assume $\text{P} = \text{NP}$ and let $B \not\in \text{NP}^*$. We want to show $B$ is $\text{NP}$-complete. We already know it is in $\text{NP}$, since $\text{NP}^* \subseteq \text{NP}$. Now we need to show it is $\text{NP}$-hard. So let $A$ be an arbitrary language in $\text{NP}$. We want to show $A \leq_p B$. Since $B \neq \emptyset$ and $B \neq \Sigma^*$ we can fix a string $Y \in B$ and a string $N \not\in B$. Also, since $A \in \text{NP}$ and $\text{P} = \text{NP}$ there is a polynomial time algorithm $M_A$ to decide $A$. Now consider the function $f: \Sigma^* \rightarrow \Sigma^*$ defined by

$$f(x) = \begin{cases} Y & \text{if } x \in A \\ N & \text{if } x \not\in A \end{cases}$$

We claim it is polynomial time computable. Indeed, a polynomial time algorithm $M_f$ to compute $f$ would take input $x$ and output $Y$ if $M_A(x)$ accepts and $N$ otherwise. (The two strings $Y, N$ are part of the description of $M_f$. Think of them as hardwired into the code.) Clearly $x \in A$ if and only if $f(x) \in B$. So $f$ is a reduction of $A$ to $B$. That is, $A \leq_p B$, as desired.