Problems:

1. Prove that there is a subset of \( \{1\}^* \) which is undecidable. (5 points)

   **ANS:** Let \( \Sigma \) be the finite alphabet of the halting problem \( H \), and \( g \) be a 1-1 and onto computable mapping from \( \Sigma^* \rightarrow \{1\}^* \).

   Then define \( G = g(H) \). Then \( G \) is the desired subset of \( \{1\}^* \) which is undecidable since \( H \) is undecidable.

2. Assume that we have two languages \( S \) and \( T \) and that \( S \) is mapping reducible to \( T \).

   (As defined in Sipser, \( S \) mapping reducible to \( T \) means that there is a (total) computable function \( f \) such that for any string \( x \), \( x \in A \) if and only if \( f(x) \in B \).)

   i. Prove that if \( T \) is decidable then \( S \) is decidable. (5 points)

   **ANS:** If \( T \) is decidable then we can decide \( S \) as follows. Given an input \( x \) (and the question is \( x \in S \)), compute \( f(x) \) and input \( f(x) \) into the decision procedure for \( T \) to decide if \( f(x) \) is in \( T \).

   If \( f(x) \) is in \( T \) then \( x \in S \) as \( f \) is a reduction, and if \( f(x) \) is not in \( T \) then \( x \not\in S \), for the same reason. So we have been able to decide if \( x \in S \) and \( S \) is decidable.

   ii. Prove that if \( T \) is recognizable then \( S \) is recognizable. (5 points)

   **ANS:** The same idea (almost). If \( T \) is recognizable then we can recognize \( S \) as follows. Given an input \( x \) (and the question is \( x \in S \)) In order to show \( S \) is recognizable we give an algorithm which accepts \( x \) if \( x \in S \), but if \( x \not\in S \) then the algorithm does not accept (it may reject or it may never halt at all).

   Our algorithm takes \( x \) and compute \( f(x) \). If then feeds \( f(x) \) into the recognizing algorithm, say \( R \), for \( T \). If \( R(f(x)) \) accepts then the algorithms accepts, if it rejects then the algorithms rejects, and if \( R(f(x)) \) never halts then the algorithm never halts.

   Clearly our algorithm accepts exactly when \( x \in S \) showing \( S \) is recognizable.

   iii. Use ii. to show that the accepting language \( A_{TM} \) is NOT mapping reducible to \( E_{TM} \), where \( E_{TM} = \{M | M \text{ is a TM and } L(M) = \phi \} \) (\( E_{TM} \) is defined in section 5.1 of the textbook.) (5 points)

   **ANS:** First recall that \( A_{TM} \) is undecidable, but it is recognizable so its complement \( \overline{A_{TM}} \) is not recognizable. And note that \( \overline{E_{TM}} \), the complement of \( E_{TM} \) is recognizable, and so since we know \( E_{TM} \) is undecidable, it is also not recognizable.

   We give a proof by contradiction. Assume it is false that \( A_{TM} \) is NOT mapping reducible to \( E_{TM} \), so \( A_{TM} \leq_m E_{TM} \).
Consider \( \overline{A_{\text{TM}}} \). Since we \( A_{\text{TM}} \) IS mapping reducible to \( E_{\text{TM}} \), we immediately get \( \overline{A_{\text{TM}}} \) is mapping reducible to \( \overline{E_{\text{TM}}} \).

Since \( \overline{E_{\text{TM}}} \) is recognizable, by part ii we have that \( \overline{A_{\text{TM}}} \) is also recognizable. But this is false, hence a contradiction.

3. i. Given a graph with \( n \) vertices, how many 3 cliques can it have (at most)? (3 points)

ANS: \( n^3 \) it may be that every 3 element sets of vertices form a triangle in the graph, ans there are \( n^3 \) of these.

ii. Show that \( L = \{G \mid G \text{ is a graph containing a 3-clique}\} \) is in \( P \). (3 points)

ANS: Just try all 3 element sets of vertices in \( G \) and check edges to see if any of them form a 3-clique in \( G \). IF yes, then accept, if no then reject. This algorithm decides if \( G \) is in \( L \) in time \( O(n^3) \).

iii. How about \( J = \{G \mid G \text{ has } n \text{ vertices and contains an } n/2\text{-clique}\} \)? Can you use the same reasoning to show \( J \) is in \( P \)? Why or why not? (3 points)

ANS: No, because the number of possible \( n/2\)-cliques in an \( n \) vertex graph is more than any polynomial, in fact it is the number \( (n \text{ choose } n/2) \) which as \( n \) gets larger grows is \( n! \).

4. Let \( \text{COMP} = \) the set of natural numbers that are not prime = \( \{ n \mid n \in \mathbb{N} \text{ and } n = ab \text{ where } a \text{ and } b \text{ are both natural number which are greater than 1} \} \).

Show that \( \text{COMP} \) is in \( NP \). (5 points)

ANS: Note that a number \( w \) is in \( \text{COMP} \) iff there exits numbers \( a \) and \( b \) between 1 and \( w \) such \( w = ab \). SO a verifier for \( V \) for \( \text{COMP} \) works as follows.

\( V \) get 3 inputs, \( w, a \) and \( b \). \( V(w,a,b) \) then multiplies \( a \) times \( b \) and checks if \( ab = w \). If \( ab = w \) then \( V(w,a,b) \) accepts and if not \( V(w,a,b) \) rejects. Noting that \( |a| \text{ and } |b| \text{ are } \leq |w| \), \( V \) is computable in polynomial time and verifies that \( \text{COMP} \) is in \( NP \).

5. Assume that you know that the clique problem is \( NP \)-complete. Use this to prove that the independent set problem is \( NP \)-complete, where \( IS = \) the independent set problem = \( \{ G,k \mid G \text{ has an independent set of size } k \} \). Note: An independent set, is a set of vertices no two of which have an edge connecting them.

Please do this carefully and with details. That is, first prove that the CLIQUE problem is in \( NP \).

Next state exactly which problem you are reducing to which and why this shows the IS problem is \( NP \)-complete. Finally give the reduction and justify (briefly) that it works.

(10 points)

ANS: Note that CLIQUE is in \( NP \) as verifier \( V \) for CLIQUE takes inputs \( G \) (the graph), \( k \) (the size of the clique), and \( C \) the set of vertices that needs to be verified as a \( k \) clique. \( V \) then checks that \( C \) has no at least \( k \) vertices from \( G \) in it, and that every edge between two vertices in \( C \) actually have an edge between them in \( G \). If the check succeeds then \( V \) accepts, otherwise it rejects. So \( V \) is a verifier for CLIQUE and CLIQUE is in \( NP \).

Similarly you can show that IS is in \( NP \) except here the verifier takes the same inputs but verifies that the set \( C \) has no edges between its vertices.
Now for the reduction (which was defined during one of the Monday lab sections). We reduce IS to CLIQUE. The reason is we want to show that any NP language is reducible to CLIQUE. We know this is true for the IS problem and that polynomial time reduction are transitive. So by reducing IS to CLIQUE we can conclude that every NP language is reducible to CLIQUE.

The reduction itself is quite simple. We define the complement graph $\overline{G}$ of G to be the graph with the same vertices as G but with edge only where G did not have edges. So $\overline{G}$ has an edge from $v$ to $w$ if G has no such edge in it.

The reduction is then let f map the pair $(G, k)$ to the pair $(\overline{G}, n-k)$ where $n =$ the number of vertices in $G$. F can be computed in polynomial time.

It is straightforward to check that $G$ has a $k$-clique if and only if $\overline{G}$ has an independent set of size $n-k$. That $f$ is a reduction follows from this fact.