CSCI 6470
Homework 6
Jiakuan Li

1.

If we can solve the Clique problem in polynomial time, so the Max Clique problem.
The algorithm is easy, we just need to keep iterate the algorithm of that solve the Clique problem.
Let $k = |G|$ at the very beginning. Keep execute the Clique problem. For each iteration, let $k = k-1$. Stop if we have a "YES" return by the Clique algorithm

2.

The previous algorithm works if the symbol in Clique problem change to "=". However, the algorithm does not work if the symbols is "$\leq$". Since both "$\geq$" and "=" sets the upper limit for Clique, so, it guarantee that $|C|$ is at least k. The "$\leq$" symbols sets the lower limit for the Clique, it guarantee that $|C|$ is at most k. So, we could not solve the Max Clique by solving the Clique Problem. Therefore, if the symbol in Clique problem changes to "$\leq$", and we could solve the Clique problem in polynomial time, we could not guarantee to solve the Max-Clique problem in polynomial time.

3.

Verifier:
Take a graph $G < V, E >$ and integer k.And a set S.
Check if $|S| \geq k$, return "NO" if false.
Check if $(u, v) \in E$ for every (u, v) pair in set S.
This verifier runs in $O(n^2)$.
Prove complete.

4.

The purpose of the polynomial reduction is going to make the problem easier. But the inverse of the reduction function usually does not hold, because these steps could not make the original problem become more difficult.

5.

Given an undirected graph $G =< V, E >$
Define a complement of G as $\overline{G} =< V, \overline{E} >$.
$\overline{G}$ has the same set of vertices as G, but $\overline{E} \cap E = \emptyset$.
If there exists a Clique V' in $\overline{G}$, the $V - V'$ is a vertex cover in G.

The reduction as follow:
1. Takes $G =< V, E >$ and integer k.
2. Generate $\overline{G}$.
3. solve Clique($\overline{G}$, k).
4. If there's a solution, return "YES".

6.

Since $L_1 \leq_P L_2$, so L_2 is at least as hard as L_1. In another word, L_1 will not be harder than L_2. Since L_2 admits an algorithm that runs in $O(n^{O(\log n)})$, so for an easier problem, L_1 should not run slower than $O(n^{O(\log n)})$. The upper bound for L_1 is $O(n^{O(\log n)})$. Therefore, if $L_1 \leq_P L_2$, and L_2 admits an algorithm runs in $O(n^{O(\log n)})$, so do L_1.

7.

Since $L' \leq_P L$, so L' is polynomial time reduction to L. This means L is at least as hard as L'. So if $L \in NP - Complete$, then $L' \in NP - Complete$ as well.