Question 1: Problem 5.22

Question 2: Problem 5.23

Question 3: Exercise 7.1 part a, b, e, f

Question 4: Exercise 7.8

Analyze this in terms of the number of nodes in G. For each step, say how much time it needs in the worse case. Justify your answers. Then give the overall time of the algorithm. To prove it is in P, simply means that it has time $O(n^c)$ for some $c$.

Question 5: Exercise 7.9

$CLIQUE$ is in NP. You are to show that the special case where $k$ is 3 is in fact in P.

Question 6: Exercise 7.6

Just do concatenation.

Give a complete proof starting with:

Let $L_1$ and $L_2$ be in P.
So, there exists deterministic Turing machines $M_1$ and $M_2$ that decide $L_1$ and $L_2$ respectively.
And $M_1$ and $M_2$ run in polynomial time.
Set $k$ so that $M_1$ and $M_2$ are in $TIME(n^k)$.

In your proof, give a high-level description of the TM, and argue why it is polynomial time.

Question 7: NP and Complementation

Part a:

The following is an incorrect proof that NP is closed under complementation. What is wrong with this proof?

Let $L_1 \in \text{NP}$.
Let $N_1$ be a polynomial time nondeterministic Turing machine that decides $L_1$.
Set $k$ so that $N_1$ is in $NTIME(n^k)$.

Construct $N$ that decides $\overline{L_1}$ as follows:

$N = $ “On input $w$

1. Run $N_1$ on $w$ 
2. If $N_1$ rejects, accept; else reject

$N_1$ is a decider and so will always halt, and will do so in $O(n^k)$. So, $N$ will also take time $O(n^k)$.

Part b:
If you have a verifier \( V_1 \) for \( L_1 \in NP \), why can’t you use this verifier to construct a verifier \( V \) for \( \overline{L_1} \)? Do not give a proof, just a rationale.

**Question 8: Exercise 7.7: Verifier**

Just do concatenation. Do this by constructing a verifier. Describe what the certificate should be, and give a high-level implementation of the verifier. Also, argue why your verifier is correct.

**Question 9: Exercise 7.7: NTM**

Just do concatenation. Let \( L_1 \) and \( L_2 \) be in NP. Show that \( L = L_1 \circ L_2 \) is in NP. For this question, construct a non-deterministic Turing machine \( N \) that accepts the \( L \). Unlike the solution to 7.15, do not make use of verifiers for \( L_1 \) and \( L_2 \), but construct \( N \) using nondeterministic TMs \( N_1 \) and \( N_2 \) that decide \( L_1 \) and \( L_2 \) respectively.

To construct \( N \) properly, make sure you do not make the same mistake as in the question titled ‘NP and Complementation’.

**Question 10: Exercise 7.5**

**Question 11: Exercise 7.18**

This is question 7.18 in the third edition, and 7.17 in the second edition.

Show that if \( P = NP \), then every language \( A \in P \), except \( A = \emptyset \) and \( A = \Sigma^* \), is NP-complete.

**Question 12: Problem 7.41**

This is question 7.41 in the third edition, and 7.39 in the second edition.

**Question 13: Problem 7.38**

This is question 7.38 in the third edition, and 7.36 in the second edition.