Question 1

By the assumption that $P = NP$, we know that $3SAT$ is in P. Remember that $3SAT$ is just a decision problem. Therefore, we only know if a formula is satisfiable or not but not necessarily knowing the satisfying assignment. However, we will show that it is possible to find the satisfying assigment by using $3SAT$ many times. Given a formula f, we use $3SAT(f)$ to check if f is satisfiable. If it is not, then we just return "unsatisfiable". Otherwise, we know there is a satisfying assignment. Now, we create a new formula f_1 by adding a new clause $\{x_1\}$ to f. such that $f_1 = f \cap \{x_1\}$. Then, we use $3SAT(f_1)$ to check if f_1 is satisfiable. It is not difficult to see f_1 is satisfiable if and only if f is satisfiable and f has at least one satisfying assignment with $x_1 = TRUE$. If f_1 is satisfiable, we don't change f_1. As we know that f is satisfiable, if f_1 is unsatisfiable, f has at least one satisfying assignment with $x_1 = FALSE$. In this case, we change $f_1 = f \cap \{\bar{x_1}\}$. At this point, we know the value of x_1 in a satisfying assigment. Then we apply the same idea to determine the value of x_2 by extending the formula f_1 as described above. So everytime we use $3SAT$, we know the value of one more variable. We use $3SAT$ at most $n+1$ times where n is the number of variables. Since $3SAT$ is in P, it is clear that the algorithm above is also in P.

Question 2

(a) Suppose that M is a polynomial time Turing machine accepting L. Then we know that for every string $x \in L$, M on x halts in state q_{accept}, and for every string $y \notin L$, M on y halts in state q_{reject}. Thus, we can create a polynomial time Turing machine M' for the complement of L by simply making the old accept state the new reject state and making the old reject state the new accept state.

(b) It does not follow that the complement of a language is also in NP. For example, $3SAT$ is in NP. THe complement of $3SAT$ consists of those strings that are not encoded properly, plus those strings that encode a $3SAT$ instance that is not satisfiable. Note that an instance f of $3SAT$ that is satisfiable has a short witness, namely, a truth assignment where f evaluates to true. But on the other hand, a formula f that is not satisfiable has no obvious short witness.

Question 3

(a) $2^{\sqrt{n}} + 5\log n = \omega(n^{\log n})$ and it is not polytime.

(b) $3^{5\log n} = O(n^10)$ and it is polytime.

(c) $8^{(7\log n)^6} = \omega(n^{\log n})$ and it is not polytime.

Question 4

(a) $3COL$ is NP-complete. You can refer the proof in lecture note 8.

(b) $UNIQUE - SAT$ is recursive but not known to be in NP. It is recursive because we can try all 2^n possible assignments to check if there is only one satisfiable assignment. However, it is not known that if $UNIQUE - SAT$ is in NP.

(c) $UCONN$ is in P. A simple DFS or BFS can do the job.

(d) $HALT - BLANK$ is not recursive. You can refer the proof in lecture note 4.

(e) $50 - CLIQUE$ is in P. You can do it by exhausively testing all subsets of vertices with size 50 if it is a clique. The complexity of the algorithm is at most $O(n^{50})$.

(f) $50 - MAXCLIQUE$ is in P. A graph has a $50 - MAXCLIQUE$ if and only if it has a $50 - CLIQUE$ but it does not have a $51 - CLIQUE$. The complexity of the algorithm is at most $O(n^5 1)$.

Question 5

The only NP-complete problem in question 4 is $3COL$. You can find the proof in lecture note 8.