Problem 1

Proof. We first show the “if” part.

Assume the enumerator, that can enumerate all the strings in the language, is $M$. We construct the following TM $M'$ that decides the language. On input string $x$, $M'$ runs $M$ to enumerate all the strings of length $|x|$. Each time $M$ enumerates a string $s$, $M'$ checks whether $x = s$. If the answer is yes, $M'$ accepts. Else, $M'$ checks whether $s$ is behind $x$ in lexical order. If yes, then $M'$ rejects. Else, $M'$ runs $M$ to enumerate the next string. If after $M$ enumerates all the strings of length $|x|$ and $M'$ still does not accept, then $M'$ rejects.

If $x$ is in the language, then $x$ must be enumerated by $M$ at some point. As $M$ enumerates every string in the language in lexical order, $M'$ will keep running and accept $x$ when $M$ enumerates it.

If $x$ is not in the language, then $M$ will never enumerate $x$. So $M'$ will reject as it cannot get a string which is enumerated by $M$ and equal to $x$.

This shows that $M'$ decides the language.

Next we show the “only if” part.

If the language is decided by a TM $M$, we construct an enumerator $M'$ that enumerates the language in lexical order. $M'$ goes through all the strings in $\{0, 1\}^*$ in lexical order. For each string $s$, $M'$ tests whether $M(s) = 1$. If yes, $M'$ outputs $s$. In this way, $M'$ enumerates all the string in the language. This is because, for every $x$ in this language, when it is tested by $M'$, $M'$ will get $M(x) = 1$ and output $x$.

Problem 2

Proof. We reduce $A_{TM}$ to $C_{TM}$.

Assume the Turing Machine $R$ decides $C_{TM}$. We construct another Turing Machine $S$ to decide $A_{TM}$. Let the input for $S$ be $\langle M, \alpha \rangle$.

We construct a Turing Machine $M_1$ and compute $R(\langle M_1, M \rangle)$. Here $M_1$ is constructed such that it only accepts the string $\alpha$, i.e. $\forall x \in \{0, 1\}^*$, if $x = \alpha$, then $M_1$ accepts; otherwise $M_1$ rejects.

$S$ accepts if and only if $R(\langle M_1, M \rangle) = 1$. 

1
Here is the explanation. First we know that $L(M_1) = \{\alpha\}$. If $M$ accepts $\alpha$, then $L(M_1) \subseteq L(M)$. So $R$ will accept. If $M$ does not accept $\alpha$, then $L(M_1)$ is not a subset of $L(M)$. So $R$ will reject.

As a result, $S$ decides $A_{TM}$. However $A_{TM}$ is undecidable. Thus $C_{TM}$ is undecidable.

**Problem 3**

*Proof.* A graph has an independent set of size at least $k$ if and only if it has an independent set of size $k$.

Let the input graph be $G = (V, E)$.

As $k$ is a constant, we can check every possible subset of vertices having size $k$ to see whether they form an independent set or not. The checking is conducted by checking that there is no edge between each pair of vertices of this subset.

There are totally $\binom{|V|}{k}$ possible subsets and checking takes time $O(k^2)$. So the overall time complexity is a polynomial of the input length.

**Problem 4**

*Solution.* (a) For any clause $C$ of $\phi$, the $\neq$-assignment will assign two literals in $C$ with different truth values. This means that one of them is 1. So $C$ is satisfied. So $\phi$ is satisfied.

For the negation of a $\neq$-assignment, every literal takes its negation. As a result, for each clause, the two literals that have different truth values are both flipped. So they are still not equal. So the assignment is still a $\neq$-assignment.

(b) Let $\phi$ be an arbitrary input for 3SAT. We transform $\phi$ to $\phi'$ according to the transformation given in the problem.

Assume $\phi$ is satisfied by some assignment. Then for any clause $c_i = (y_1 \lor y_2 \lor y_3)$, at least one of $y_1, y_2, y_3$ is 1 under this assignment. In order to satisfy $(y_1 \lor y_2 \lor z_i)$ and $(\bar{z}_i \lor y_3 \lor b)$, we can let $z_i = \neg(y_1 \lor y_2)$ and $b = 0$. We can see that this is a $\neq$-assignment because $z_i = \neg(y_1 \lor y_2)$ and $b = \neg(z_i \lor y_3)$. So $\phi'$ has a $\neq$-assignment.

On the other hand, assume $\phi'$ has a $\neq$-assignment. If $b = 1$ in this assignment, by (a) we can take the negation of the assignment to get another $\neq$-assignment which has $b = 0$. If $b = 0$ in the assignment, then for each clause $c_i$, the literals $y_1, y_2, y_3$ cannot be all 0 (otherwise no $z_i$ can simultaneously satisfy $(y_1 \lor y_2 \lor z_i)$ and $(\bar{z}_i \lor y_3 \lor b)$). So $c_i$ is satisfied and thus $\phi$ is satisfied.
This shows that $\neg$SAT is NP-hard.

(c) We need to show that $\neg$SAT is in NP. For any input string $\phi$, the certificate is an assignment $x$ to the variables. We know that $\phi \in \neg$ SAT if and only if there exists a $\neg$-assignment. We can check that $x$ is a $\neg$-assignment by checking that each clause has two literals that have different values. This can be done in polynomial time. So $\neg$SAT is in NP.

As a result, $\neg$SAT is NP-complete. \qed