Problem 1

Answer. 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 $M(\langle M \rangle) = 1$. If yes, $M'$ outputs $s$. In this way, $M'$ enumerates all the strings in the language. This is because, for every $x$ in this language, when it is tested by $M'$, $M'$ will get $M(\langle x \rangle) = 1$ and output $x$.

\[ \square \]

Problem 2

Answer. We reduce the function UC to $T$.

Assume we have a TM $R$ that decides $T$. Now we are going to construct another TM $S$ that computes UC.

$S$ works in the following way. Assume the input is $\langle M \rangle$. Then $S$ first constructs another TM $M_1$ as follows.

Assume the input for $M_1$ is $x$.

- If $x \neq 001$ then $M_1$ accepts.
- Else, i.e., $x = 001$, $M_1$ computes $M(\langle M \rangle)$ and accepts if and only if $M(\langle M \rangle) = 1$. (Note: if $M(\langle M \rangle)$ does not halt then $M_1(\langle x \rangle)$ does not halt either.)

$S$ uses $R$ to get $R(\langle M_1 \rangle)$ and sets $S(\langle M \rangle) = 1 - R(\langle M_1 \rangle)$.

Here is the explanation. If the input $\langle M \rangle$ is such that $M(\langle M \rangle) = 1$, $R$ accepts $\langle M_1 \rangle$ as $M_1$ accepts every string. So $S(\langle M \rangle) = 0$. On the other hand, if $M(\langle M \rangle) = 0$ or $M$ does not halt on $\langle M \rangle$, then $M_1$ does not accept $x = 001$. This means that $M_1$ accepts every string except 001. So $\langle M_1 \rangle$ is not in $T$. Thus $R(\langle M_1 \rangle) = 0$ and $S(\langle M \rangle) = 1$.

According to the definition of UC, $S$ computes UC correctly. However we know that UC is not computable. This means that $T$ is not decidable.

\[ \square \]
Problem 3

Answer. 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 $R((M_1, M))$. 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 $R((M_1, M)) = 1$.

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 4

Answer. We give an algorithm $A$ which can decide 2COL in polynomial time.

Assume the two colors are cyan and violet. Given a graph $G$, $A$ first chooses an arbitrary vertex $u$ and colors it cyan. Without loss of generality, we assume $G$ is a connected graph (if not, we can run the algorithm on all its connected components). Then $A$ conducts a depth first search starting at $u$. Once $A$ visits a vertex $v$ through an edge $e = (w, v)$, it tests whether $v$ has already been colored. If yes, $A$ tests whether $w$ and $v$ have different colors. If they have the same color, $A$ rejects. If $v$ has not been colored, then $A$ will color it with the color different from $w$. $A$ keeps doing the searching until every vertex is colored. If every vertex is colored and $A$ still does not reject, $A$ accepts. The running time of the algorithm is linear in the input size.

Now we prove the correctness of our algorithm.

If $G$ is in 2COL, then all the circles in $G$ have even number of vertices, because if a circle, having an odd number of vertices, be colored by two colors, there must be two adjacent vertices having the same color. As a result, when we run $A$, it will not reject. This is because, if $A$ rejects on visiting a vertex $v$, the circle $u \rightarrow v \rightarrow u$ has an odd number of vertices, which is a contradiction. So $A$ will accept at last.

On the other hand, if $G$ is not in 2COL, our algorithm will reject. Towards a contradiction, suppose our algorithm accepts at last. This means that every vertex is colored either cyan or violet. Also no two adjacent vertices are colored by the same color. This is because, assume there is a pair of vertices $u$ and $v$ which are two ends of the edge $e = (u, v)$ and have the same color. We know that one of them is colored first in our algorithm. Assume it is $u$. Then after $A$ colored $v$, $A$ will visit $u$, finding that $u$ and $v$ have the same color, and reject. As a result, the graph is 2-colored which contradicts that $G$ is not in 2COL. So our algorithm will reject.

This shows the correctness of our algorithm.

Problem 5

Answer. According to the definition of NP, as $L_1$ is in NP, there exists a polynomial time TM $M_1$ and a polynomial $p_1 : \mathbb{N} \rightarrow \mathbb{N}$ such that for every $x \in \{0, 1\}^*$,

$$x \in L_1 \Leftrightarrow \exists u \in \{0, 1\}^{p_1(|x|)} \text{ s.t. } M_1(x, u) = 1.$$  

Also, as $L_2$ is in NP, there exists a polynomial time TM $M_2$ and a polynomial $p_2 : \mathbb{N} \rightarrow \mathbb{N}$ such that for every $x \in \{0, 1\}^*$,

$$x \in L_2 \Leftrightarrow \exists u \in \{0, 1\}^{p_2(|x|)} \text{ s.t. } M_2(x, u) = 1.$$
(a) The language $L_1 \cup L_2$ is in $\text{NP}$. This can be shown by constructing another verifier $R$ which is a TM that works as follows. On input $x$ and $u = u_1 \circ u_2 \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$ ($u_1$ has length $p_1(|x|)$), $R$ computes $M_1(x, u_1)$ and $M_2(x, u_2)$. If at least one of them outputs 1, $R$ outputs 1. Otherwise $R$ outputs 0.

We notice that $R$ runs in polynomial time as both $M_1$ and $M_2$ run in polynomial time.

If $x \in L_1$, let $u'_1$ be such that $M_1(x, u'_1) = 1$. Then $R$ will output 1 on input $\langle x, u'_1 \circ 1^{p_2(|x|)} \rangle$. If $x \in L_2$, let $u'_2$ be such that $M_2(x, u'_2) = 1$. Then $R$ will output 1 on input $\langle x, 1^{p_1(|x|)} \circ u'_2 \rangle$. Thus if $x \in L_1 \cup L_2$, there is a $u \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$ such that $R(x, u) = 1$. On the other hand, if $x \notin L_1 \cup L_2$, we know that $\forall u_1 \in \{0, 1\}^{p_1(|x|)}$, $M_1(x, u_1) = 0$ and $\forall u_2 \in \{0, 1\}^{p_2(|x|)}$, $M_2(x, u_2) = 0$. So $\forall u \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$, $R(x, u) = 0$. This proves that there exists a polynomial time TM $R$ s.t.

$$x \in L_1 \cup L_2 \iff \exists u \in \{0, 1\}^{p_1(|x|)+p_2(|x|)} \text{ s.t. } R(x, u) = 1.$$  

So $L_1 \cup L_2$ is in $\text{NP}$.

(b) The language $L_1 \cap L_2$ is also in $\text{NP}$. It can be shown by constructing another verifier $R$ which is a TM that works as follows. On input $x$ and $u = u_1 \circ u_2 \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$ ($u_1$ has length $p_1(|x|)$), $R$ computes $M_1(x, u_1)$ and $M_2(x, u_2)$. If both of them output 1, $R$ outputs 1. Otherwise $R$ outputs 0.

As a result, first we know that $R$ runs in polynomial time as both $M_1$ and $M_2$ runs in polynomial time. Second, if $x \in L_1 \cap L_2$, there exists $u_1 \in \{0, 1\}^{p_1(|x|)}$ such that $M_1(x, u_1) = 1$ and there exists $u_2 \in \{0, 1\}^{p_2(|x|)}$ such that $M_2(x, u_2) = 1$. So there is a $u = u_1 \circ u_2 \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$ such that $M_1(x, u_1) = 1$ and $M_2(x, u_2) = 1$. Thus $R(x, u) = 1$. On the other hand, if $x \notin L_1 \cap L_2$, we know that either $\forall u_1 \in \{0, 1\}^{p_1(|x|)}$, $M_1(x, u_1) = 0$ or $\forall u_2 \in \{0, 1\}^{p_2(|x|)}$, $M_2(x, u_2) = 0$. So $\forall u \in \{0, 1\}^{p_1(|x|)+p_2(|x|)}$, $R(x, u) = 0$. Thus there exists a polynomial time TM $R$ s.t.

$$x \in L_1 \cap L_2 \iff \exists u \in \{0, 1\}^{p_1(|x|)+p_2(|x|)} \text{ s.t. } R(x, u) = 1.$$  

So $L_1 \cap L_2$ is in $\text{NP}$.

\[\square\]

**Remark 1.** The proof can also be done using non-deterministic Turing machines, since the two definitions of $\text{NP}$ are equivalent.