Problem 1-1 (TM’s with Infinite States)  

Argue that every language $L \subseteq \Sigma^*$ can be decided by a Turing machine with countably infinite number of states. Be sure to formally define all 7 components of the corresponding TM (most importantly, the transition function).

(Hint: Make sure you get the “little details” right, like where the end of the input is.)

Problem 1-2 (Other Models of TM’s)  

Show the equivalence of the following variant of TM’s with the usual TM’s. You are allowed to use $k$-tape TM’s if you wish. In this variant, a TM has a “doubly infinite tape” (i.e., a tape which is infinite in both directions, so that the machine can always move to both left and right). The tape is initially filled with blanks, except the input is placed directly to the right of the initial head position. Your simulation can be reasonably “high level”, there is no need to go into “state diagram”, as long as you are reasonably convincing.

If a TM with doubly infinite tape will halt in time $t$, how many steps will the equivalent ordinary TM take (for this estimation, go to the single-tape TM if a multi-tape TM was used for simulation; you can use without proof that a $k$-tape TM halting in time $T$, where $k \geq 2$, can be simulated by an ordinary TM in time $O(T^2)$). For an extra credit, try to achieve linear-time simulation. Namely, your ordinary machine should halt in time $O(t)$.

Problem 1-3* (Lexicographic Enumerators)  

Recall, a language is recursively enumerable iff it has an enumerator machine (the one that starts with an empty tape, and eventually prints out all the strings in the language, possibly many times). Show that the language is recursive iff it has an enumerator which prints the strings of the language exactly once and in the lexicographic order. Using this result show that every infinite recursively enumerable language contains an infinite recursive sublanguage.

(Hint: You may wonder if the problem deserves the “star”. Well, it does. Think of the empty language and your proof to see why.)

Problem 1-4 (Context Sensitive Grammars)  

Recall, a context-free grammar $\Gamma = \{V, \Sigma, S, R\}$ has variables $V$, terminals $\Sigma$, start symbol $S$ and transition rules $R$, plus the restriction that every rule in $R$ maps some variable $X \in V$ to an arbitrary string $\beta \in (V \cup \Sigma)^*$. In a context-sensitive grammar, we can map an arbitrary non-empty string $\alpha \in (V \cup \Sigma)^*$ to some $\beta \in (V \cup \Sigma)^*$, as long as $|\alpha| \leq |\beta|$ (with a possible exception for the rule
$S \to \epsilon$, in which case $S$ is not allowed on the right hand side of any rule). As usual, the language accepted by such $G$ is the set of all strings $s \in \Sigma^*$, which could be obtained from the start symbol $S$. Any such language is called context-sensitive.

(a) Show that every context-sensitive language is recursive.

(Hint: Use a multi-tape TM. Argue that every string $s \in L(G)$ must have a “finite” derivation and use “brute force” from there on. Lemmas 5.7 and 5.8 in Sipser might provide good intuition for this problem. Indeed, it turns out that the class of context-sensitive languages is exactly the class of languages accepted by an LBA. You don’t have to show it, of course, but it might provide some good analogy.)

(b) Let $\langle G \rangle$ stands for a string representing the description of a grammar. Let

$$L = \{ \langle G \rangle \mid G \text{ is a context-sensitive grammar and } \langle G \rangle \notin L(G) \}$$

Show that $L$ is recursive but not context-sensitive.

(Hint: Do exactly what was done to show that $A_{TM}$ was undecidable.)

Problem 1-5 (Halting in 33 steps) 20 points

Formulate each of the following problems about properties of TM’s as a language and place it in its appropriate class ($R$, $RE\setminus R$, $coRE\setminus R$ or $RE \cup coRE$). Be concise: try to prove the minimal number of assertions, i.e. do not use the same argument several times.

(a) $M$ halts on some input within at most 33 steps.

(Hint: How many configurations can $M$ reach in 33 steps?)

(b) $M$ halts on all inputs within at most 33 steps.

(Hint: Same hint as in (a).)

(c) $M$ halts on some input within more than 33 steps.

(Hint: Is this the complement of (b)? Be careful.)

(d)* (optional, 10 points extra credit) $M$ halts on all inputs within more than 33 steps.

Problem 1-6 (Ambiguity of CFG’s) 10 points

Let $AMBIG_{CFG} = \{ \langle G \rangle \mid G$ is a an ambiguous CFG$\}$ (recall, a grammar is ambiguous if it has two different leftmost derivations of some string). Using the following mapping reduction from the Post Correspondence Problem (PCP), show that $AMBIG_{CFG}$ is undecidable (i.e., prove that the reduction below is indeed a mapping reduction). Given an instance of PCP

$$P = \left\{ \left[ \frac{t_1}{b_1} \right], \left[ \frac{t_2}{b_2} \right], \ldots, \left[ \frac{t_k}{b_k} \right] \right\}$$

construct a CFG $G$ with the rules

$$S \to T \mid B$$

$$T \to t_1 T a_1 \mid \ldots \mid t_k T a_k \mid t_1 a_1 \mid \ldots \mid t_k a_k$$

$$B \to b_1 B a_1 \mid \ldots \mid b_k B a_k \mid b_1 a_1 \mid \ldots \mid b_k a_k$$

where $a_1 \ldots a_k$ are new terminal symbols.
Problem 1-7 (Useless States) 20 points

Given an NFA, PDA or NTM, we say that it has a useless state if some of its states is never reached, on any input and any non-deterministic branch! Define the classes \(\text{USELESS}_C = \{ (M) \mid M \in C \text{ and has a useless state} \} \), where \( C = \text{NFA, PDA, NTM} \).

(a) Show \(\text{USELESS}_{\text{PDA}}\) is recursive. If this is too hard, do the same for a simpler language \(\text{USELESS}_{\text{NFA}}\).

(Hint: Recall that \(E_{\text{PDA}} = E_{\text{CFG}}\) is recursive. Try to use this to design a test if a given state is useless. Then try all states.)

(b) Show \(\text{USELESS}_{\text{NTM}}\) is not recursively enumerable.

(Hint: Reduce its complement to the halting problem.)

Problem 1-8 (Reductions) 15 points

(a) Give an example of an undecidable language \(C\) such that \(C \leq_m \overline{C}\). Can you find such \(C \in \mathcal{R}E\setminus\mathcal{R}\)?

(Hint: Let \(C = \{2x \mid x \in \ldots\} \cup \{2x + 1 \mid x \in \ldots\}\). You have to fill the dots...)

(b) For any languages \(A\) and \(B\) construct a language \(C\) such that \(A \leq_m C\) and \(B \leq_m C\).

(Hint: Let \(C = \{2x \mid x \in \ldots\} \cup \{2x + 1 \mid x \in \ldots\}\). You have to fill the dots...)

(c) Extra Credit: Construct Turing-incomparable languages \(A\) and \(B\) (i.e., \(A \nleq_t B\) and \(B \nleq_t A\)).

Problem 1-9 (New Proof of Rice’s Theorem) 15 points

Using recursion theorem, give an alternative proof of Rice’s theorem. For that, use function

\[
h(M,x) = \begin{cases} 
A(x), & \text{if } M \text{ is a TM in } R_{\Gamma} \\
B(x), & \text{otherwise}
\end{cases}
\]

where \( \Gamma \subset \mathcal{RE} \) is a non-trivial property of \( \mathcal{RE} \) languages and \( R_{\Gamma} = \{ (M) \mid L(M) \in \Gamma \} \) is the claimed undecidable property of TM’s. You have to guess what \( A \) and \( B \) are...

(Hint: Recall what it means for \( \Gamma \) to be “non-trivial”.)

Extra 1* (Recursion Theorem Live!) 20 points

In the C language, write the shortest program you can that will output its own description. Please do not copy it from the book or someone else, it’s fun to do it on your own! Name your program “your last name”.c and hand in the transcript with the execution of the following commands: 1) cat name.c, 2) wc name.c, 3) gcc name.c -o name.out, 4) ./name.out. The person with the shortest program will get even more extra credit!