12. Given that L is Turing recognizable, some enumerator E enumerates L. Consider another TM D that operates as follows. For a given input i, D runs E until it prints $< M_i >$, the ith item. Then D runs M_i on input i. Finally, if M_i accepts, D rejects, and if M_i rejects, D accepts. Clearly, D always terminates on all inputs, and is thus a decider itself. Let's call the (decidable) language of D as L_D.

Now, let's assume that there is some $< M > \in L$ such that M decides L_D. Then $< M >$ must appear at some index k in the enumeration done by E. Now, since M and D are both deciders for L_D, M must accept input k iff D accepts input k. However, by definition, D rejects k if M accepts k. Therefore, our supposition that such a TM M exists is wrong, which implies that the language L_D has no decider in L. This concludes the proof.

20. Consider the following transformation procedure on any DFA D.

1. Create an NFA N^r as follows.
2. Copy over all states from D into N^r.
3. If there exists a transition $\delta_D(q, a) \rightarrow q'$ in D, add correspondingly the transition $\delta_{N^r}(q', a) \rightarrow q$. This is visually equivalent to reversing the transition edges.
4. Add a new state s and mark it as a start state of N^r. Add transitions of the form $\delta_{N^r}(s, \epsilon) \rightarrow f$ for each state f that is an accept state in D.
5. Make the start state of D the only accept state of N^r.
6. Finally, convert the NFA N^r into an equivalent DFA D^r.

Now, we can see from the above construction that N^r (and thus also D^r) accepts w^r iff D accepts w. Therefore, N^r (and thus also D^r) accepts both w^r and w iff D too accepts both w and w^r. Taking this to its conclusion, we can see that N^r (and thus also D^r) and D are equivalent iff for any string w, D accepts w^r iff it accepts w.

Now, consider a TM M that for any given DFA description $< d >$, performs the above DFA transformation procedure on d, and then checks whether d and d^r are equivalent. The transformation procedure itself takes a finite number of steps. Further, we know (from chapter 4 text) that the problem of determining whether two DFAs are equivalent is decidable, and thus can also be done in a finite number of steps. Therefore the TM M is a decider. In fact, it decides the language $S = \{< D > \mid D \text{ is a DFA that accepts } w^r \text{ iff it accepts } w\}$. This concludes the proof.

22. Given that A and B are co-Turing recognizable, there are recognizers for $\overline{A}$

and $\overline{B}$. Call these M and N. Consider a TM T that operates as follows. For a given input w, it runs M and N one step at a time taking turns, and accepts if N accepts first, else rejects. Because A and B are disjoint, either M or N terminates on any given input. Therefore T is a decider. Call the language of T as C.

Now, we need to prove two things. Firstly, we need to prove that $A \subseteq C$. Consider the execution of T on any input $w \in A$. In that execution, N accepts the input and M rejects that input (or loops forever). Finally, T accepts the input, and thus $w \in C$. Therefore $A \subseteq C$.

Secondly, we need to prove that $B \subseteq \overline{C}$. Consider the execution of T on any input $w \in \overline{B}$. In that execution, N accepts the input. If it accepts first, T accepts and $w \in C$ else T rejects and $w \notin C$. Therefore, $C \subseteq \overline{B}$. Since A and B are disjoint, this implies that $B \subseteq \overline{C}$.

This implies that the decidable language C (decided by T) divides A and B, thus concluding the proof.

24. There are two directions to prove. Firstly, say we have a decidable language D such that $C = \{x \mid \exists y(< x, y >\in D)\}$. Then there must be some decider for D. Call it M_D. We'll use this to construct a recognizer M_C for C that operates as follows. For a given input x, M_c enumerates all strings in lexicographic order and for each such string y invokes M_D on the input $< x, y >$. This is guaranteed to terminate if $x \in C$. Therefore, M_C is a recognizer for C, and as such C is a Turing recognizable language.

Secondly, say that C indeed were a Turing recognizable language. Then, there must be some enumerator E_C that enumerates C. We'll use this to construct a decider M_D that operates as follows. For a given input $< x, y >$, M_D runs E_C for upto y steps, then checks if E_C printed x, and finally accepts if it did, and rejects if it did not. Clearly, M_D is a decider since it terminates on all inputs. Further, it decides a language D such that $C = \{x \mid \exists y(< x, y >\in D)\}$. This concludes the proof.