1. Carefully describe (give state diagram) a TM which will add one to the binary representation of a number. The number will have a $ on the left end.
   - If the input is the empty string, then the output should be $.
   - If the input is $, the output should be $0
   - If the input is (for example) $1010, the output should be $1011, and $111 should result in $1000
   - Leading zeroes are acceptable ($010 becomes $011)
   - After correctly transforming the input, halt by entering the accepting state
   - If the input is poorly formed (such as $$ or $01$0), reject it.

2. exercise 3.13: What can a Turing machine with stay-put instead of left compute?

3. exercise 4.30: Let $A$ be a Turing-recognizable language consisting of descriptions of Turing machines $\{\langle M_1 \rangle, \langle M_2 \rangle, \ldots \}$, where every $M_i$ is a decider. Prove that some decidable language $D$ is not decided by any decider $M_i$ whose description appears in $A$. (Hint: you may find it helpful to consider an enumerator for $A$.)

4. (grads) exercise 4.17 (2nd ed) or 4.18 (3rd ed): Let $C$ be a language. Prove that $C$ is Turing-recognizable if and only if a decidable language $D$ exists such that

$$C = \{ x \mid \exists y \ (\langle x, y \rangle \in D) \}.$$ 

Note: In the text this is a starred (difficult) problem. It should not be, and is important in understanding the Turing-recognizable ($\equiv$ recursively enumerable) languages. It has also an important analogy in the characterization of $NP$.

Hint (for $\Rightarrow$): Think of $y$ as the number of steps for which to simulate the TM for $C$. 