Problem 1. Draw a TM $M$ which decides the language $A = \{a^n b^n c^n | n \geq 0\}$, where the input alphabet $\Sigma = \{a, b, c\}$.

Solution 1. The principle of this Turing machine is to sweep right while checking for an $a$, $b$, and $c$ in that order without breaking the formatting. We mark the ones we find until we run out of one type. When we do, if we did not run out of $a$’s first, then we reject. If we did run out of $a$’s first, we need only check that we’re also out of $b$’s and $c$’s.

For the sake of clarity, all missing transitions in the Turing machine below go to the rejecting state.

Problem 2. Give an algorithm for a TM $M$ which decides the language

$$C = \{\langle D, \omega \rangle | D \text{ is a DFA and } D \text{ accepts } \omega \}.$$  

Note that $\langle D, \omega \rangle$ is some reversible encoding of $D$ and $\omega$ into a string. The precise details of how this encoding is computed is irrelevant to the algorithm you give. The key point is you have access to each parameter (you should do more than write ”Simulate $D$ on an input”).
Solution 2. Since a DFA is not entirely compatible with Turing machines (it doesn’t know its own input, for example), we need to give a little more information than just use the universal Turing machine $U$ to simulate $D$ on $\omega$. We will give a slightly more verbose answer than is strictly necessary for the sake of completeness.

Define the Turing machine $M$ to be

$M = \text{On input } \nu$
1. If $\nu \neq \langle D, \omega \rangle$, then reject
2. Let $D = (Q, \Sigma, \delta, q_0, F)$
3. Let $\omega = \omega_1 \ldots \omega_{|\omega|}$
4. Let $q = q_0$
5. For $i = 1$ to $|\omega|$
   (a) Compute $\delta(q, \omega_i) = q'$
   (b) Assign $q = q'$
6. Accept if $q \in F$ and reject otherwise

Each step of $M$ is computable and is guaranteed to complete, so $M$ halts. Moreover, $M$ exactly simulates $D$ on $\omega$, so $L(M) = C$.

Problem 3. Explain why the class of decidable languages is closed under

a) union,
b) intersection,
c) concatenation,
d) Kleene star.

You do not need to formally prove each, but your explanation should sketch the outline of a proof.

Lastly, describe how your explanations would differ for the class of recognizable languages.

Solution 3.

a) Given two recognizable languages $A$ and $B$, there are TMs $M_A$ and $M_B$ which recognize them. We can construct a new TM $M$ which recognizes $A \cup B$ by simulating $M_A$ and $M_B$ in parallel. $M$ accepts an input $\omega$ if either of $M_A$ or $M_B$ accepts $\omega$.

b) Given two recognizable languages $A$ and $B$, there are TMs $M_A$ and $M_B$ which recognize them. We can construct a new TM $M$ which recognizes $A \cap B$ by simulating $M_A$ and $M_B$ in parallel. $M$ accepts an input $\omega$ if both $M_A$ and $M_B$ accept $\omega$.

c) Given two recognizable languages $A$ and $B$, there are TMs $M_A$ and $M_B$ which recognize them. We can construct a new TM $M$ which recognizes $AB$ by simulating $M_A$ and then $M_B$. $M$ accepts an input $\omega$ if we can split $\omega$ into two strings $\omega = xy$ such that $M_A$ accepts $x$ and $M_B$ accepts $y$. If there is no such split, we reject.
d) Given a recognizable language $A$, there is a TM $M_A$ which recognizes it. We can construct a new TM $M$ which recognizes $A^*$ as follows.

On input $\omega$, $M$ should first see if $M_A$ accepts $\omega$. If not, $M$ should split $\omega$ into two strings in every possible way. If $M_A$ accepts both strings of any split of $\omega$, it should accept. We repeat this process for splitting $\omega$ into three strings, then four, and so on until we split $\omega$ into its individual symbols. If $M_A$ accepts none of these splits, we reject.

The difference in these proofs if we were dealing with the recognizable languages is that we would need to use dovetailing to simulate the Turing machines described above (only for concatenation and Kleene star). This is because while the deciders are guaranteed to halt on any input, recognizers are not.

**Problem 4.** Define $\omega_i$ to be the $i^{th}$ element of $\Sigma^*$ in lexicographical order. When $\Sigma = \{0, 1\}$, for example, this order is $\epsilon$, 0, 1, 00, 01, 10, 11, 000, etc...

To prove that enumerators are equivalent to Turing machines, we created an enumerator $E$ given a TM $M$ such that $L(E) = L(M)$. The algorithm for this is given below.

$E =$ "Ignore input,

1. For $i = 1$ to $\infty$
   a. For $j = 1$ to $i$
      i. Run $M$ on input $\omega_j$ for $i$ steps
      ii. If $M$ accepts, print $\omega_j"

Explain why the following algorithm for $E$ does not work.

$E =$ "Ignore input,

1. For $i = 1$ to $\infty$
   a. Run $M$ on input $\omega_i$
   b. If $M$ accepts, print $\omega_i"$

**Solution 4.** The trouble with the second algorithm is that $M$ is not guaranteed to halt on any particular string. Suppose $M$ loops forever on string $\omega_i$ but there is some $\omega_j$ with $j > i$ that $M$ accepts (it is trivial to show such an $M$ exists). Then $E$ will never output $\omega_j$.

**Problem 5.** Let $E$ be an enumerator that prints its language in lexicographical order. Prove that $L(E)$ is decidable.

**Solution 5.** We give a TM $M$ which decides $L(E)$.

$M =$ On input $\omega$,

1. Run $E$
   a. $E$ prints a string $\nu$
   b. If $\nu = \omega$, accept
   c. If $\nu$ comes after $\omega$ lexicographically, reject

2. Reject

Clearly, if $E$ prints a string, $M$ must accept it eventually. Moreover, if $E$ does not print a string, it must eventually either halt or print a string that occurs after an input $\omega$ to $M$. In either case, $M$ will reject. As such, $M$ always halts and $L(M) = L(E)$. 

3