Answer the questions below, paying particular attention to the logic of your arguments. Short descriptions for how a function might computed (on any machine we've talked about) are sufficient to show that it is computable. Pseudocode is often appropriate for reductions.

1. Show that if A is recognizable and A reduces to $\bar{A}$, then A is decidable.

By definition, a language L is decidable if and only if L and its complement $\bar{L}$ are recognizable. We are given that A is recognizable, so all that is left is to show that $\bar{A}$ is recognizable. From the definition of reduction, if A is a reduction of B and B is recognizable, then A is recognizable as well. In this case, $\bar{A}$ is a reduction of A and A is recognizable, so $\bar{A}$ is recognizable as well. So we have shown that A and $\bar{A}$ are both recognizable, so A is decidable.

2. Let $B = \{< M > \mid M$ accepts exactly one of the strings 00 and 11, i.e. $|L(M) \cap \{00,11\}| = 1\}$.
    a. What does Rice's theorem say about B?

Rice's Theorem states conditions for when a language is undecidable. This language does not meet one of those conditions. There can exist two machines that recognize this language B but are still both in B. Such machines would be simple to construct. One machine only accepts '00' and the other only accepts '11.' As such, this language may be decidable.

    b. Show that halting problem ($H_{TM} = \{< M > \mid M('')$ halts $\}$) reduces to B.

```
Def R(<M>):
   Def N(x):
        M('')
        Return x == '00'
   Return <N>
```

This reduction works because if M('') halts, then we return a string description of a machine that only accepts '00' (and so |L(M) intersection {00,11}| is 1). If M('') does not halt, we return nothing, which is a string description of a machine that accepts nothing (and so the size of the set intersection is 0).

    c. Show that halting problem reduces to the complement of B.

```
Def R(<M>):
   Def N(x):
        If x == '00':
             M('')
             Return true
        Else if x == '11':
             Return true
        Else:
```

```
            Return false
    Return <N>
```

This reduction works because if M(‘’) does halt, we accept ‘00’ and ‘11’ (and thus $|L(M) \cap \{00,11\}| = 2$. If M(‘’) does not halt, the only input we accept is ‘11’ and so the size of the intersection is 1.

d. Are B or its complement recognizable?

The halting problem might loop, but that is acceptable for recognizability. In this case, since B and its complement are reductions of the halting problem, they too are recognizable by the properties of reduction.

3. A language is *co-recognizable* if its complement is recognizable. Argue why each of the following languages is or is not recognizable and why it is or is not co-recognizable.
    a. $L_1 = \{< M > | M$ enters state q_{27} for the input string $001101\}$.

L_1 is recognizable. By definition of decidable, the machine must accept <M> if it enters state q_{27} but can reject or loop on other input. This language is recognizable because the machine can accept if state q_{27} is reached. If the machine halts and state q_{27} was not reached, then <M> is not in L_3. If M loops before reaching state q_{27}, then <M> is not in L_3. All these cases fulfil the requirements of recognizability.

The complement of L_1 (the string representations of machines such that the machine does *not* enter state q_{27} for input 001101) is also recognizable. A Turing machine can be constructed such that it halts for input 001101 and does not enter state q_{27}. As such, the complement of L_1 is recognizable.

Since the complement of L_1 is recognizable, L_1 is co-recognizable.

    b. $L_2 = \{< M > | L(M)$ contains at most two strings$\}$.

L_2 is not recognizable. To show that M accepts at most two strings, all the strings that can be created with the alphabet of M would have to be tested. While this is countably infinite, it is still infinite and cannot be computed even with strategies such as dovetailing.

The complement of L_2 is recognizable. All that must be shown is that the machine M accepts more than two strings, which can be accomplished by feeding input into the machine in a paralleled dovetail fashion. Once three strings have been accepted, we know that <M> is in the complement of L_2.

Since the complement of L_2 is recognizable, L_2 is co-recognizable even though it is not recognizable.

    c. $L_3 = \{< M > | L(M) \subseteq \Sigma^*\}$

L_3 is recognizable. It is the string descriptions of Turing machines such that the Turing machine accepts something. This is similar to the complement of L_2 above. To recognize L_3, the Turing machine M can be

simulated and fed inputs in a parallel dovetail fashion. As long as one of the input strings is accepted, <M> is in L_3.

The complement of L_3 rejects everything. To show that a Turing machine M is in the complement of L_3, every possible input string would have to be feed to the machine and none of them could be accepted. It is not possible to simulate every input combination, and as such, the complement of L_3 is not recognizable.

Since the complement of L_3 is not recognizable, L_3 is not co-recognizable.

4. A computable verifier is a deterministic Turing machine V that takes two arguments: x (the input) and y (the proof). A computable verifier always halts. Show that a language L is recognizable if and only if there exists a computable verifier V such that
    a. if $x \in L$, then there is a string y such that V(x,y) accepts, and
    b. if $x \notin L$,then V(x,y) rejects for every string y.

First, we need to show that if L is recognizable, a computable verifier exists. From the definition of recognizability, we know that for all strings *x* that are elements of language L, there exists a Turing machine M such that M accepts x. For strings that are not elements of L, M may loop or reject, but not accept. In this case with verifier V(x,y), we let V simulate the machine M that accepts L and define y to be the number of steps that V is allowed to execute. We can see that a y exists such that it prevents V from looping infinitely but provides sufficient steps that all the strings that are elements of L will be accepted. If V does not accept before y steps, we assume that it is looping and reject x. So we have shown that if L is recognizable, computable verifier V exists with the specified properties.

Second, we need to show that if a computable verifier V with the above properties exists for language L, than L is recognizable. Since we are assuming we have a verifier, we can create a machine M that uses V to recognize L. V has to accept all strings within language L with at least one proof y, but has to reject all strings not in L no matter the proof y that is used. For this reason, we cannot define y to be the number of steps V is allowed to run as we did above (because the set of natural numbers will not cover every proof). Instead, we have to define y to be a proof represented in a (potentially infinite) binary string, i.e., $y \in \{0,1\}^*$. If x is in language L, recognizer M passes new proofs y to verifier V until V accepts at which point M accepts. If string x is not in language L, recognizer M will pass new proofs to V, but V will never accept, so M will loop forever, thus not accepting input x. This meets the requirements for recognizability and thus L is recognizable.

5. Consider the following property: $P = \{L \mid L$ is the language of some Turing machine that has an odd number of states$\}$. Show that *P*is a trivial property. (Yes, this problem is trivial).

By applying Rice's theorem, we can see that this is a trivial property. If P does not meet the requirements of Rice's theorem, then P is trivial. In this case, we can construct two machines that recognize the same language L but that have a different number of states. In other words, one machine has an even number of states and the other machine has an odd number of states, but both recognize the same language. In fact,

turning one machine into the other is not difficult. All that is necessary is to add an "orphan" state (a state that is not reachable, so it will not affect the computation) and one machine turns into another.

[Bonus] Consider the property $P = \{L \mid L$ is accepted by some Turing machine that never halts after an even number of steps$\}$. Show that *P*is a trivial property.

We can apply Rice's theorem again here. If the property does not meet the conditions of Rice's theorem, then it is a trivial property. In this case, it would once again be possible to construct two machines that recognize the same language, but have one be in P and one not be in P. In this case, you could simply add a state that does nothing before each accept and reject state. Thus, every machine in P is turned into a machine not in P, even though the two machines accept the same language. This fails the first requirement of Rice's theorem, and so P is a trivial property.

6. [Bonus] Read about The Recursion Theorem in the Sipser text. One implication of the recursion theorem is that in any general purpose programming language, one can write code that outputs the code itself. Write a python program that prints its own code. Do not use any file operations.

https://www.udacity.com/course/viewer#!/c-ud557/l-1209378918/m-2986218594