1. We will use cantor's diagonalization method to show that the set β consisting of all infinitely long sequences consisting of the digits 0 and 1 is uncountably infinite.

   First we assume that β is countable. Then we know every element in the set can be listed. We will list them in such a way below. using arbitrarily chosen random sequences from the set.

Sq_1	1	1	0	1	1	0	1	0	...
Sq_2	1	0	1	0	0	1	0	0	...
Sq_3	1	1	1	1	0	0	0	0	...
Sq_4	1	0	0	1	1	0	0	1	...
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

   Now, consider if we take the 'red' numbers from the set and flip all of the 1's to 0's and 0's to 1's. and continue this pattern down the infinitely long diagonal line of red digits. We can then construct some sequence Sq_s composed of these digits. Their order in Sq_s is the order they appear in the enumeration table if you are reading diagonally from top-left to bottom-right.

   We will undoubtedly change exactly one element of every sequence within our presumably complete, countable set. Since this is the case then even as the number of sequences in the list extends to infinity, we will always be able to find some new sequence of 1's and 0's different from the sequences listed– namely Sq_s. If this is the case then the list is not complete and we have a contradiction because an incomplete list is uncountable, Thus the set β is uncountably infinite and therefore uncountable.

2. To prove set T is countable it is sufficient to show that there is a bijective correspondence f mapping from T to some subset of the natural numbers.

   We can easily show this correspondence by constructing a way to list every element in T.

   We know each element in T is a tuple consisting of exactly 3 members of the set of Natural Numbers. We can then enumerate these elements by simply concatenating the three members. That is f(t) = i · j · k. where t is a tuple in T, i is the first member of t, j, is the second, and k is the third. Because i,j, and k are Natural numbers by the definition of T, it follows that i · j · k is also a natural number.

   the following table shows what we mean. consider three random tuples of the infinitely large set T

   (42, 4, 2)
   (42, 1, 5999)
   (10, 68,209,389,842, 42)

T_i = t	t- i	t- j	t- k	$f(t)$
(42, 4, 2)	42	4	2	4,242
(42, 1, 5999)	42	1	5999	4,215,999
(10, 68,209,389,842, 42)	10	68,209,389,842	42	106,820,938,984,242

By constructing a list in such a way we have shown that for every unique element in T there is a mapping to a unique element of the set of Natural Numbers.

It then follows that the range of f(T) is some subset of the set of Natural Numbers. By Cantor's definition of size and using the principle that any subset of some countable set is also countable, we have proved that T is itself countable.

---

3. In order to show that a the language S is decidable it is sufficient to show we can build a Turing Machine TM_S which decides it. The approach for this will include checking to see if the input ¡M¿ is a DFA in S, then TM_S will accept. But if the input is not a DFA in S, then reject

   Proof by Construction:

   Let TM_S be the Turing machine which decides the language consisting of all DFAs which upon acceptance of some string w will also accept the reverse of w.

   To accomplish this functionality we can make use of the fact that if a DFA accepts w and accepts w^R then the same DFA will also accept W^R and w, because if a word leads to an accept state when being iterated from left to right and it also leads to an accept state when it is being iterated in reverse, then we can conclude that iterating it in reverse and iterating it forwardly will both lead to accept states as well.

   Therefore, TM_S can determine if the input ¡M¿ is in S, by reversing the transitions of ¡M¿ into a different DFA called N^R and seeing if N^R and M are equal.

   TM_S works as follows.

   1. Enumerate every possible transitional path in DFA M from the start state to every accept state.

   2. Construct a different NFA M^R which reverses the transitions of each of these paths. (eg. if M transitions from state "Start" to state "1" on a and from "1" to "2" on b and from "2" to "accept" on a, then M^R will perform the reverse: from state "accept" to "2" on a, from "2" to "1" on b and from "1" to "start" on a.)

   3. switch the start state to be the accept state and swap the accept state(s) to be nondeterministic start state(s) for M^R.

   4. convert M^R into an equivalent DFA D^R using Theorem 1.39 from Sipser's text.

   5. Simulate the Turing Machine EQ_{DFA} from Theorem 4.5 of Sipser's text on input ¡M, D^R ¿

   (if M is in S, then D^R will be identical to M for every path from start to every accept).

   6. If EQ_{DFA} accepts, accept. If EQ_{DFA} rejects, reject.

This machine TM_S will effectively decide if the input provided is a DFA in the language S. Therefore we have proved S is decidable. ———

4. In order to show that a the language F is decidable it is sufficient to show we can build a Turing Machine TM_F which decides it. The approach for this will include checking to see if the input ¡P¿ when in state ¡q¿ and ¡x¿ is on top of the stack is in a looping state, then TM_F will accept. But if the input does not succeed to be in a looping state, then reject

   Proof by Construction:

   Let TM_F be the Turing machine which decides the language consisting of all PDAs which upon coming to state some state with some character on top of the stack will never see the element below that character or read in any more input.

   To accomplish this functionality we can make use of TM_F's knowledge of whether or not the next transition will read a character from the input or not.

   TM_F works as follows.

   1. Realize the transition function of PDA ¡P¿ when in state ¡q¿ and reading epsilon from input and peeking at character ¡x¿ from the top of the stack.

   2. Write TM_F's the encoding for state q and character ¡x¿ on top of the stack

   3. For each possible transition from the current configuration, non-deterministically simulate P, all the while checking to see if P's state and top of stack ever match the encoding and character written on the tape. If they do, then stop simulating that branch of non-determinism.

   4. If any of these transitions read a character from input or reach an accept state. TM_F accepts immediately. otherwise, if all the possible transition paths have been exhausted, then reject.

   This machine TM_F will effectively decide if for some P q and x, a looping situation occurs. Therefore we have proved F is decidable because some Turing machine can decide it.

   ———-

5. EXTRA CREDIT

   We shall show that the class of decidable languages is not closed under homomorphism by utilizing a a counter example.

   Then let L be a Decidable Language. where h is a homomorphism on its alphabet. Then L' = h(L) = {h(w) | w is in L }

   Now consider the example where there are an infinite number of words in L.

   Imagine now that some of the output of h(L) is an infinitely long sequence of characters.

   Therefore L' consists of infinitely many variations of infinitely long strings.

We know from class that no Turing machine can decide a language with infinitely many variations of infinitely long strings because the possible computations are enumerable. (ie. Cantor's diagonalization method shows the necessary cases to consider are uncountable. if they are uncountable than they can't be programmed.)

So if L' is not decidable, then we have proved by counter example that the class of decidable languages is not closed under homomorphism.