1. To show that a "Doubly Infinite Tape Turing Machine" recognizes the class of Turing Recognizable Languages we can simply show that it can be used to simulate a ordinary Turing Machine.

   This can be accomplished in two steps.

   1.Upon initialization move left one space and write some symbol not in Σ to mark the beginning, (if the tape is transitioning from right to left), of the tape which is not to be accessed– that is: the portion of the tape which doesn't exist in an ordinary Turing Machine. Then transition to the right one space.

   2. If the machine ever reads that symbol, immediately transition to the right one space, moving the head back to the first character of the input string.

   In performing these steps only one space of the extra tape is used. That space is then used to carry out the exact same function as the left-hand end of the tape of an Ordinary Turing Machine. Therefore the "Doubly Infinite Tape Turing Machine" will certainly recognize the same class of languages as ordinary Turing Machines because it can be modified to behave in an identical manner.

   ---

2. To show that a Turing Machine S which cannot move left back over the input does not recognize the class of Turing Recognizable Languages it is sufficient to provide an example of a Turing Recognizable Language which is not recognizable by S.

   Consider the language A = $\{0^{2^n} | n \geq 0\}$.

   Without the ability to move back left over the string after crossing off every other "two" as the touring machine which recognizes this language must be able to do, a Turing Machine S is not capable of recognizing such a language.

   This proves S does not recognize the class of Turing Recognizable Languages as required.

   The class of languages S recognizes is some proper subset of Regular Languages, because without the ability to move left and without any form of memory (other than the ability to write to the current position on the tape, which it can never return to after leaving) there is no way it can zig-zag in order to compare or push onto a stack in order to remember the number of occurrences of two different characters. At best it can recognize only languages which specify those sequences of characters in the recognized language which have no variations in the number of occurrences of each character– that is: there are two possible types of languages. One type requires an exact range of occurrences to be specified by which the range can not be dependent on any other variable which could change depending on the input, and the other type requires there be no specified number of occurrences at all, only the order of characters within all accepted sequences within the language.

   ---

3. 3.15 //

   (b)

   In order to show that the concatenation of two Turing-Decidable languages from the set of all Turing-Decidable languages$\{l_1, l_2, ...l_i\}$ produces a a Turing-Decidable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can decide it.

   We can construct a Turing Machine T3 which consists of the two Turing Machines which decide the substrings of the concatenation.

   We can take the Turing Machine T1 which decides the language from which the left-hand side of the concatenation is of, and run it on the input tape from the first character of the input up to every instance where it decides either accept or reject.

   Simultaneously we run the second Turing Machine T2 which decides the language from which the right-hand side of the concatenation is of starting at the first character of the input, and then restarting non-deterministically on every character which led T1 to decision.

   In doing so we guarantee the concatenation of two Turing-Decidable languages is also a Turing Decidable Language, because we have constructed a Turing Machine which decides it. thus Turing-Decidable languages are closed under concatenation.

   In order to show that the Kleene Star of a Turing-Decidable language from the set of all Turing-Decidable languages$\{l_1, l_2, ...l_i\}$ produces a a Turing-Decidable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can decide it.

   We can construct a Turing Machine T3 which closely resembles the Turing Machine T1 which decides l_i

   We add a transition which accepts the empty string if one does not already exists.

   Every time we reach a state where T1 would have accepted should it have been the end of the input, we program T3 to accept if it is the end of the input, or if it is not to rewrite the remaining input characters onto another tape and non-deterministically treat that tape like it is an initial input tape.

   In doing so we guarantee the Kleene Star operator on a Turing-Decidable language produces yet another Turing Decidable Language, because we have constructed a Turing Machine which decides it. thus Turing-Decidable languages are closed under Star operation.

   In order to show that if the complementation operator is applied to a Turing-Decidable language from the set of all Turing-Decidable languages$\{l_1, l_2, ...l_i\}$ then produced is a Turing-Decidable language L, it is sufficient to show that for all variations of L there is a Turing Machine which can decide it.

We can construct a Turing Machine T3 which closely resembles the Turing Machine T1 which decides l_i

We simply apply the transition functions from T1, and if the T1 would have rejected, we accept; if T1 would have accepted we reject.

Every time we reach a state where T1 would have accepted should it have been the end of the input, we program T3 to accept if it is the end of the input, or if it is not to rewrite the remaining input characters onto another tape and non-deterministically treat that tape like it is an initial input tape.

In doing so we guarantee the complemenation operator on a Turing-Decidable language produces yet another Turing Decidable Language, because we have constructed a Turing Machine which decides it. thus Turing-Decidable languages are closed under complementation.

In order to show that the intersection of two Turing-Decidable languages from the set of all Turing-Decidable languages$\{l_1, l_2, ...l_i\}$ produces a Turing-Decidable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can decide it.

We can construct a Turing Machine T3 which consists of the two Turing Machines $T_i and T_j$ which decide the two languages supplied to the intersection of $l_i and l_j$. We can simulate this combined machine by simply reading the input with both machines.

We run the transition function of Turing Machine T_i on the input tape and see if it accepts or rejects, if T_i accepts then we move the head back to the left-hand side of the tape and apply T_j's transition function to it. if Both Machine's accept the input, then T3 will also accept the input, else T3 will reject.

In doing so we guarantee the intersection of two Turing-Decidable languages is also a Turing Decidable Language, because we have constructed a Turing Machine which decides it. thus Turing-Decidable languages are closed under intersection.

---

4. 3.16

(b)

In order to show that the concatenation of two Turing-Recognizable languages from the set of all Turing-Recognizable languages$\{l_1, l_2, ...l_i\}$ produces a a Turing-Recognizable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can recognize it.

We can construct a Turing Machine T3 which consists of the two Turing Machines which recognize the substrings of the concatenation.

We can take the Turing Machine T1 which recognizes the language from which the left-hand side of the concatenation is of, and run it on the input tape from the first character of the input up to the first character which it does not recognize.

Simultaneously we run the second Turing Machine T2 which recognizes the language from which the right-hand side of the concatenation is of starting at the first character of the input, and then restarting non-deterministically on every character which led T1 to recognition.

In doing so we guarantee the concatenation of two Turing-Recognizable languages is also a Turing Recognizable Language, because we have constructed a Turing Machine which recognizes it. Thus Turing-Recognizable languages are closed under concatenation.

———————————————————————————————-

(c)

In order to show that the Kleene Star of a Turing-Recognizable language from the set of all Turing-Recognizable languages$\{l_1, l_2, ...l_i\}$ produces a a Turing-Recognizable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can recognize it.

We can construct a Turing Machine T3 which closely resembles the Turing Machine T1 which recognizes l_i

We add a transition which accepts the empty string if one does not already exists.

Every time we reach a state where T1 would have accepted should it have been the end of the input, we program T3 to accept if it is the end of the input, or if it is not to rewrite the remaining input characters onto another tape and non-deterministically treat that tape like it is an initial input tape.

In doing so we guarantee the Kleene Star operator on a Turing-Recognizable language produces yet another Turing Recognizable Language, because we have constructed a Turing Machine which recognizes it. Thus Turing-Recognizable languages are closed under Star operation.

———————————————————————————————-

(d)

In order to show that the intersection of two Turing-Recognizable languages from the set of all Turing-Recognizable languages$\{l_1, l_2, ...l_i\}$ produces a Turing-Recognizable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can recognize it.

We can construct a Turing Machine T3 which consists of the two Turing Machines $T_i and T_j$ which recognize the two languages supplied to the intersection of $l_i and l_j$. We can simulate this combined machine by simply reading the input with both machines.

We run each Turing Machine T_i and T_j on the input tape and see if either recognizes the input, we then move the head back to the left-hand side of the tape and apply the the other

Turing Machine to it. if Both Machine's recognize the input, then T3 will also recognize the input.

In doing so we guarantee the intersection of two Turing-Recognizable languages is also a Turing Recognizable Language, because we have constructed a Turing Machine which recognizes it. Thus Turing-Recognizable languages are closed under intersection.

——————————————————————————-

(e)

In order to show that the homomorphism operator is applied to a Turing-Recognizable language from the set of all Turing-Recognizable languages$\{l_1, l_2, ...l_i\}$ produces a a Turing-Recognizable language L. It is sufficient to show that for all variations of L there is a Turing Machine which can recognize it.

We can construct a Turing Machine T3 which closely resembles the turing Machine T1 which recognizes T_i

We then simply amend the transition statement of T1 to include the transition which formerly went from some character x to some character y to be a transition from the last character of each newly acceptable substring to y. everything else is the same.

if the new transition leads to an accept state then T3 will accept, else it will reject.

In doing so we guarantee the homomorphism of two Turing-Recognizable languages is also a Turing Recognizable Language, because we have constructed a Turing Machine which recognizes it. Thus Turing-Recognizable languages are closed under homomorphism.

——————————————————————————-

5. EXTRA CREDIT

   To show that an infinite Turing-Recognizable language R has an infinite Turing-Decidable subset D we will go to the definition and Theorem 3.21 from Sipser's book//

   we know that if L is Turing Recognizable, then from Theorem 3.21 there is an enumerator which enumerates it. We also know that if L is infinite, then the enumerator must not loop or halt as the enumerator prints every word in the language. Therefore if the enumerator is capable of printing an infinite number of strings, then there is an infinite set of strings in L we will call set I. The set of I contains an infinite number of printed words which are in the language L. If I is an infinite set of words in L and each word in I can be enumerated, then it can also be decided, because each word printed must have been accepted first. Therefore there exists a Turing-Decidable subset of L; proved as required.