1. To show that $L$ is decidable, we construct a TM $M$ that does the following on input $w$:

1. Mark the first tape cell with a distinct mark that denotes the beginning of the tape.
2. Scan $w$ for an unmarked $b$; if an unmarked $b$ is found, mark it, move left to the beginning of the tape, and go to the next step. If no $b$ is found, check for an unmarked $a$. If no unmarked $a$ is found, then reject; otherwise, accept.
3. Scan $w$ to find an unmarked $a$; mark it if one is found go to the next step; otherwise no unmarked $a$ was found, so accept.
4. Continue scanning $w$; if another unmarked $a$ is found, mark it, move left to the beginning of the tape, and go to step 2; otherwise a second unmarked $a$ was not found, so accept.

This machine will look for a $b$ and try to find two $a$s for each $b$ it sees. Since at each step of the search, the machine knows whether it has seen two $a$s for each $b$ or not, it will always be able to halt with the correct answer. Therefore, $L$ is decidable.

2. To show that $L$ is decidable, we construct a TM $M$ that does the following:

1. On input $\langle A, B \rangle$, where $A$ and $B$ are DFAs, construct a DFA $C$ with $L(C) = L(A) \cap L(B)$ by using De Morgan’s laws:
   1. Construct DFAs $A'$ and $B'$ that recognize $\overline{L(A)}$ and $\overline{L(B)}$ by swapping the accepting and non-accepting states.
   2. Construct an NFA $C'$ with $L(C') = L(A') \cup L(B')$.
   3. Obtain a DFA $C''$ by performing the subset construction on $C'$.
   4. Construct a DFA $C$ with $L(C) = \overline{L(C'')}$.
2. Let $E$ be the Turing machine that decides $E_{DFA}$. Run $E$ on $\langle C \rangle$.
3. If $E$ accepts, then accept. If $E$ rejects, then reject.

Since $E$ decides $E_{DFA}$, it is guaranteed to halt and give an answer. Thus, our Turing machine is guaranteed to halt and give an answer. Thus, this machine decides $L$ and $L$ is decidable.

3. (a) Given a Turing machine $M$ that decides a language $L$, we can construct a Turing machine $M'$ which decides $\overline{L}$. $M'$ operates as follows:

1. On input $w$, run $M$ on $w$. 


2. If \( M \) rejects \( w \), then \textit{accept}; if \( M \) accepts \( w \), then \textit{reject}.

Since \( L \) is decidable, \( M \) is guaranteed to halt. Thus \( M' \) decides \( \overline{L} \) and \( \overline{L} \) is decidable.

(b) Let \( M_1 \) and \( M_2 \) be Turing machines that recognize languages \( L_1 \) and \( L_2 \), respectively. We can construct a Turing machine \( M_3 \) that recognizes \( L_1 \cap L_2 \). \( M_3 \) operates as follows:

1. On input word \( w \), run \( M_1 \) on \( w \). If \( M_1 \) accepts, then go to the next step. If \( M_1 \) rejects, then \textit{reject}.

2. Run \( M_2 \) on \( w \). If \( M_2 \) accepts, then \textit{accept}; otherwise, \textit{reject}.

First, we note that \( M_3 \) accepts \( w \) only if both \( M_1 \) and \( M_2 \) accept \( w \) and \( M_3 \) will reject \( w \) if at least one of \( M_1 \) or \( M_2 \) rejects \( w \). However, if either \( M_1 \) or \( M_2 \) do not halt, then \( M_3 \) does not halt. Thus, \( M_3 \) recognizes \( L_1 \cap L_2 \).

(c) Let \( M_1 \) and \( M_2 \) be Turing machines that recognize languages \( L_1 \) and \( L_2 \), respectively. We can construct a Turing machine \( M_3 \) that recognizes \( L_1 \cdot L_2 \). \( M_3 \) operates as follows:

1. On input \( w \), nondeterministically split \( w \) into two parts \( w = w_1w_2 \).

2. Run \( M_1 \) on \( w_1 \). If \( M_1 \) accepts, then go to the next step. If \( M_1 \) rejects, then \textit{reject}.

3. Run \( M_2 \) on \( w_2 \). If \( M_2 \) accepts, then \textit{accept}. If \( M_2 \) rejects, then \textit{reject}.

We note that \( M_3 \) will only accept \( w \) if there exists a branch of computation where \( w_1 \) is accepted by \( M_1 \) and \( w_2 \) is accepted by \( M_2 \). If there is no \( w_1 \) that is accepted by \( M_1 \), \( M_3 \) either halts and rejects or runs forever. The same applies to \( M_2 \) if there exists some \( w_1 \) that is accepted by \( M_1 \) but no suitable \( w_2 \) is accepted by \( M_2 \).

4. To show that a doubly-infinite Turing machine can simulate an ordinary TM, we simply mark the initial tape cell with a special symbol that disallows the machine from moving to the left of the cell.

To show that an ordinary Turing machine can simulate a doubly-infinite Turing machine, instead, we show that a 2-tape TM, which we have shown to be equivalent in power to the ordinary TM, can simulate a doubly-infinite tape. Let \( D \) be a doubly-infinite TM and let \( M \) be our 2-tape TM. We split the tape of \( D \) into two parts and assign each part to a tape of \( M \). Tape 1 of \( M \) corresponds to the part of the tape of \( D \) that contains the input word and everything to the right. Tape 2 of \( M \) contains everything on the tape of \( D \) to the left of the input word in reverse order.

More formally, let \( w_0 \) denote the contents of the tape cell that contained the first symbol of the input word at the beginning of the computation of \( D \). At the beginning of computation, \( M \) must mark the leftmost cell of each tape with some symbol \# so
the machine can tell where it needs to switch tapes. Then if $D$ has a tape $uw_0v$, Tape 1 of $M$ contains $#w_0v$ and Tape 2 of $M$ contains $#u^R$.

5. We show that $FIN(\Sigma)$ has a correspondence with the set of binary words $\{0, 1\}^*$, which we know to be countable. We also know that the set of words over $\Sigma$ is countable and can be enumerated in lexicographic order $s_1, s_2, s_2, \ldots$. We define the characteristic string of a language $L \in FIN(\Sigma)$ to be a binary string $b = b_1 b_2 \cdots b_n$ with

$$b_i = \begin{cases} 
0 & \text{if } s_i \notin L, \\
1 & \text{if } s_i \in L.
\end{cases}$$

If $s_n$ is the lexicographically greatest string in $L$, then we define $s_j = \varepsilon$ for all $j > n$. The string $s_n$ must exist since $L$ is finite. Then every finite language $L$ has a finite characteristic binary string and every finite binary string corresponds to a finite language over $\Sigma$. Thus, $FIN(\Sigma)$ is countable.