Exercise 9.3.5

Let \( L \) be the language described in problem, we will first show that it is a RE, and then show that it is not recursive.

- **\( L \) is RE**
  
  We need to construct a TM \( M \) to accept \( L \). The idea is to simulate \( M_1 \) and \( M_2 \). However, we do not know whether \( L(M_1) \) or \( L(M_2) \) are recursive or not, so we cannot simply enumerate every string and feed it into \( M_1 \) and \( M_2 \) to see whether it is accepted or not. A feasible way is to enumerate the upper bound of running time of \( M_1 \) and \( M_2 \). We enumerate \( i = 1, 2, \ldots, \) and for every \( i \), execute \( M_1 \) and \( M_2 \) on all the strings \( s_1, s_2, \ldots, s_i \) each for at most \( i \) steps. There is also a counter in \( M \), keeping the number of different strings that are accepted by \( M_1 \) and \( M_2 \). When the counter reaches \( k \), \( M \) halts and accepts \( (M_1, M_2, k) \). If \( |L| \geq k \), TM \( M \) will definitely halt in finite time and answer “yes”. So \( L \) is RE.

- **\( L \) is not recursive**
  
  We want to reduce \( L_{\text{ne}} \) to \( L \). Given any TM \( M \), such that \( L(M) = L \), we can construct a TM \( M' \) as follows:
  
  The input to \( M' \) is a code of some TM \( M'' \), we feed \( (M'', M'', 1) \) into the given TM \( M \). If \( M \) accepts \( (M'', M'', 1) \), then \( M' \) accepts \( M'' \); if \( M \) rejects, then \( M' \) rejects; if \( M \) runs forever, \( M' \) runs forever. It is easy to see that \( M' \) is indeed a TM for \( L_{\text{ne}} \). If \( L \) were recursive, we could guarantee \( M \) to halt on any input, which suggested that \( M' \) were guaranteed to halt, and therefore \( L_{\text{ne}} \) were recursive. Here comes the contradiction, since \( L_{\text{ne}} \) is not recursive. So we conclude that \( L \) is not recursive.