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 accept $(M_1, M_2, k)$. 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 $L_{ne}$ 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 for ever, $M'$ runs for ever. It is easy to see that $M'$ is indeed a TM for $L_{ne}$. If $L$ were recursive, we could guarantee $M$ to halt on any input, which suggested that $M'$ were guaranteed to halt, and therefore $L_{ne}$ were recursive. Here comes the contradiction, since $L_{ne}$ is not recursive. So we conclude that $L$ is not recursive.

Comment:

- Generally, we need to show a language is RE in a “constructive” way. It is not correct to show $L$ is RE by showing that $\overline{L}$ is not RE. There exists some language that is not RE and its complementary is not RE either, although such an example is non-trivial.

- After constructing a TM $M$ to accept $L$ (hence proving that $L$ is RE), we cannot directly claim that $L$ is not recursive by showing that $M$ might not terminate. We need to know that any possible TM for $L$ will not be guaranteed to terminate, not for a particular TM.

- If we want to use reduction to prove a result about hardness, the correct way is to reduce from a known set to the one we want to prove, but not the way around. Our intuition of solving an unknown problem is to reduce it to some known problem, but it is not the case here.