1. a) This language is decidable. If l is greater than the pumping length p, then we check to see if $\langle M \rangle \in ALL_{DFA}$. If so, then $\langle M \rangle \in$ LorBIGGER$_{DFA}$. This is because all strings larger than the pumping length will eventually loop down to strings within the pumping length. We know ALL_{DFA} is decidable.
Now, if $0 \leq l \leq p$ then we check to see if all strings from length l to p are in the language of M. If they are then we make sure that there are no transitions from a state further than l transitions away from the start state to a state that is within l steps from the start state that has a path to a reject state. If so, then $\langle M \rangle \in$ LorBIGGER$_{DFA}$.

d) If a Turing Machine rejects a finite number of inputs, then it belongs to BIG-ENUF$_{TM}$ as l will equal one more than the length of the largest rejected string. However, if a Turing Machine rejects infinitely many inputs then no such l exists. Now, Consider the computable function

$$\begin{aligned} f(w) &= w \text{ if w is not a valid TM description} \\ &= M' \text{ if } w = \langle M \rangle \end{aligned}$$

where M' is a machine identical to M except its reject state leads to a sink state where all future inputs with the same prefix are rejected.
This function shows that $ALL_{TM} \leq_m$ BIG-ENUF$_{TM}$ because if M accepts all input, then M' will behave identically to M and accept all inputs. This leads to $l = 0$ as all strings over $\Sigma^*_{M'}$ will be accepted, so $\langle M' \rangle \in$ BIG-ENUF$_{TM}$. On the other hand, if M rejects some input, M' will reject all inputs with that as a prefix, making the number of rejected inputs infinite so $\langle M' \rangle \notin$ BIG-ENUF$_{TM}$.
At the same time, f shows that $\overline{ALL_{TM}} \leq_m \overline{\text{BIG-ENUF}_{TM}}$
.
We know that neither ALL_{TM} nor $\overline{ALL_{TM}}$ is recognizable. Therefore, we can conclude that neither BIG-ENUF$_{TM}$ nor $\overline{\text{BIG-ENUF}_{TM}}$ is recognizable.

2. a) Assume that $P = NP$ for the sake of contradiction. We know that $L \in P$ given that $L \in NP$. This means that an instance of L can be verified in polynomial time and that there is some polynomial-time algorithm for solving it. Since L is not NP-complete then we know an NP-complete problem cannot be reduced to L. This means that an instance of L cannot be solved in polynomial time to solve an instance of any NP-complete problem. However, this violates our statement that there is some polynomial-time algorithm for solving L, so our assumption that $P = NP$ is incorrect. Therefore, if there exists an $L \in NP$ that is not NP-complete then it must be the case that $P \neq NP$.