Proving that a language \( L \) is undecidable by reduction requires several steps:

- Choose a language \( L' \) that you already know is undecidable. Typical choices for \( L' \) include:

  \[
  \begin{align*}
  \text{ACCEPT} & := \{ (M, w) \mid M \text{ accepts } w \} \\
  \text{REJECT} & := \{ (M, w) \mid M \text{ rejects } w \} \\
  \text{HALT} & := \{ (M, w) \mid M \text{ halts on } w \} \\
  \text{DIVERGE} & := \{ (M, w) \mid M \text{ diverges on } w \} \\
  \text{NEVERACCEPT} & := \{ (M) \mid \text{ACCEPT}(M) = \emptyset \} \\
  \text{NEVERREJECT} & := \{ (M) \mid \text{REJECT}(M) = \emptyset \} \\
  \text{NEVERHALT} & := \{ (M) \mid \text{HALT}(M) = \emptyset \} \\
  \text{NEVERDIVERGE} & := \{ (M) \mid \text{DIVERGE}(M) = \emptyset \}
  \end{align*}
  \]

- Describe an algorithm (really a Turing machine) \( M' \) that decides \( L' \), using a Turing machine \( M \) that decides \( L \) as a black box. Typically this algorithm has the following form:

  Given a string \( w \), transform it into another string \( x \), such that \( M \) accepts \( x \) if and only if \( w \in L' \).

- Prove that your Turing machine is correct. This almost always requires two separate steps:
  - Prove that if \( M \) accepts \( w \) then \( w \in L' \).
  - Prove that if \( M \) rejects \( w \) then \( w \notin L' \).

Prove that the following languages are undecidable:

1. \( \text{ACCEPTILLINI} := \{ (M) \mid M \text{ accepts the string } \text{ILLINI} \} \)
2. \( \text{ACCEPTTHREE} := \{ (M) \mid M \text{ accepts exactly three strings} \} \)
3. \( \text{ACCEPTPALINDROME} := \{ (M) \mid M \text{ accepts at least one palindrome} \} \)