1) \( \text{ALLONE}_{DFA} \) is decidable. Idea: If \( 1^* \subseteq L(M) \), then there is no string from \( 1^* \) in \( \overline{L(M)} \) so we can equivalently check the emptiness of the language \( 1^* \cap \overline{L(M)} \).

Proof: The following TM \( T \) decides \( \text{ALLONE}_{DFA} \).

\[ T = \text{“On input} \langle M \rangle \text{ where } M \text{ is a DFA,} \]

- Construct the DFA \( D \) so that \( L(D) = \overline{L(M)} \cap 1^* \) (regular languages are closed under compliment and intersection).
- Run TM \( E \), a decider for \( E_{DFA} \) on input \( \langle D \rangle \).
- If \( E \) accepts, accept; if \( E \) rejects, reject.”

2) \( \text{ALLONE}_{PDA} \) is decidable. Idea: We can turn a given PDA \( P \) into a CFG \( G \) in CNF and then run \( A_{CFG} \) on every string \( 1^k \) for \( 0 \leq k \leq p + p! \), where \( p \) is at least as large as the pumping length of \( L(G) \). We can obtain such a number by the same method described in Theorem 2.34 (Proof of the Pumping Lemma for CFLs) so \( p = 2^{1/2} + 1 \) where \( V \) is the set of variables in \( G \). Now, if every \( 1^k \) for \( 0 \leq k \leq p + p! \) is found to be generated by \( A_{CFG} \), we then know that for any \( 1^j, j > p + p! \), some string in \( \{1^p, \ldots, 1^{p+p!}\} \) can be pumped to get \( 1^j \). This is because we can let \( c = (j-p)\%p! + p \), \( \% \) represents the mod operator which is greater than the pumping length \( p \) so can be pumped. Now, no matter what number \( Num_{ones} \) of ones (note this number is between \( 1 \) and \( p \)) is pumped, \( 1^j \) is yielded by pumping since the difference between \( j \) and \( c \) is a multiple of \( p! \) and thus is a multiple \( Num_{ones} \).

Proof: The following TM \( T \) decides \( \text{ALLONE}_{PDA} \).

\[ T = \text{“On input} \langle M \rangle \text{ where } M \text{ is a PDA,} \]

- Construct a grammar \( G \) in Chomsky Normal Form that is equivalent to \( M \) (by methods in Lemma 2.27 and Theorem 2.9).
- Run TM \( A \), a decider for \( A_{PDA} \) on input \( \langle G, 1^k \rangle \) for every \( 0 \leq k \leq p + p! \).
- If \( A \) accepts every string, accept; if \( A \) rejects, reject.”

3) \( \text{ALLONE}_{TM} \) is undecidable. Idea: If there is a decider \( R \) for \( \text{ALLONE}_{TM} \), we can create a decider for \( A_{TM} \), which we know is impossible so it must be that \( \text{ALLONE}_{TM} \) is undecidable.

Proof: Assume \( \text{ALLONE}_{TM} \) is decidable so that there is a decider \( R \) for \( \text{ALLONE}_{TM} \). The following TM \( A \) decides \( A_{TM} \).

\[ A = \text{“ input} \langle M, w \rangle \text{ where } M \text{ is a TM and } w \text{ is a string from } \Sigma^* \]

- Create TM \( H \) defined as follows:
  - “On input \( x \),
    - if \( x \) is not of the form \( 1^k \), reject \( x \)
    - otherwise, run \( M \) on input \( w \) and if \( M \) accepts, accept \( x \)” (note, we cannot say “otherwise reject \( x \)” since we do not know if \( M \) will terminate)
- Run \( R \) on \( \langle H \rangle \). If \( R \) accepts, accept \( \langle M, w \rangle \), otherwise reject \( \langle M, w \rangle \).”

Now \( L(H) = \emptyset \) if \( M \) does not accept \( w \) and \( L(H) = 1^* \) if \( M \) accepts \( w \). Thus, since \( R \) can decide if \( \langle H \rangle \) is in \( \text{ALLONE}_{TM} \), this corresponds to deciding if \( M \) accepts \( w \) so that \( A \) is a decider for \( A_{TM} \), a contradiction so \( \text{ALLONE}_{TM} \) is undecidable.

Proof 2: We can use Rice’s Theorem if we can show that this is a non-trivial property of TMs. To do this, we need to show that the property is a property of the language and some, but not all, TMs have this property. Since Turing machines can accept regular languages, we can use the TMs that accept \( \emptyset \) and \( 1^* \)
as the evidence that some, but not all, TMs have this property. Second, if \( L(M_1) = L(M_2) \) for two turing machines \( M_1 \) and \( M_2 \), then it is certainly the case that the property is true for both or neither since any subset of \( L(M_1) \) is a subset of \( L(M_2) \). Thus, Rice’s Theorem applies so \( \text{ALLONE}_{TM} \) is undecidable.

4) ASSIGN is undecidable. Idea: We can show that a decider for ASSIGN enables us to make a decider for \( \text{HALT}_{TM} \).

Proof: Assume that ASSIGN is decidable. Let \( R \) be a decider for ASSIGN. The following TM \( T \) is a decider for \( \text{HALT}_{TM} \):

\[
T = \text{"On input } \langle M, w \rangle \text{ where } M \text{ is a TM and } w \text{ is a string from } \Sigma^* \text{"
}
\begin{itemize}
  \item Create a C program that simulates \( M \) and at the end of the program, add an explicit assignment to NEWVAR, a new variable not already occurring in the program and call this program \( P \).
  \item Run \( R \) on \( \langle P, \text{NEWVAR}, w \rangle \).
\end{itemize}

If \( M \) halts on input \( w \), then \( P \) assigns \( \text{NEWVAR} \) an explicit value. However, since \( \text{HALT}_{TM} \) is undecidable, this is a contradiction so ASSIGN is undecidable.

5a) \( L \) is not Turing-recognizable. Idea: If there were a recognizer for \( L \), we could make a recognizer for \( \overline{A_T M} \).

Proof: Let \( R \) be a recognizer for \( L \). The following TM \( T \) is a recognizer for \( \overline{A_T M} \):

\[
T = \text{"On input } \langle M, w \rangle \text{ where } M \text{ is a TM and } w \text{ is a string from } \Sigma^* \text{"
}
\begin{itemize}
  \item Run \( R \) on \( \langle 1, M, w \rangle \).
  \item If \( R \) accepts, accept.”
\end{itemize}

Since the input to \( R \) starts with a 1, \( \langle 1, M, w \rangle \in L \) if \( \langle M, w \rangle \in \overline{A_T M} \). Since \( \overline{A_T M} \) is not Turing-recognizable, this is a contradiction so \( L \) is not Turing-recognizable.

5b) \( L \) is not co-Turing-recognizable. Idea: If there were a recognizer for \( \overline{L} \), we could make a recognizer for \( \overline{A_T M} \).

Proof: Let \( R \) be a recognizer for \( \overline{L} \). The following TM \( T \) is a recognizer for \( \overline{A_T M} \):

\[
T = \text{"On input } \langle M, w \rangle \text{ where } M \text{ is a TM and } w \text{ is a string from } \Sigma^* \text{"
}
\begin{itemize}
  \item Run \( R \) on \( \langle 0, M, w \rangle \).
  \item If \( R \) accepts, accept.”
\end{itemize}

Since the input to \( R \) starts with a 0, \( \langle 0, M, w \rangle \in \overline{L} \) if \( \langle M, w \rangle \notin A_T M \) which is equivalent to \( \langle M, w \rangle \in \overline{A_T M} \). Since \( \overline{A_T M} \) is not Turing-recognizable, this is a contradiction so \( L \) is not co-Turing-recognizable.