1. A. Algorithm written in python

function PL-TRUE?(s, m) # s is the sentence and m is symbol truth model

if len(s) == 0: # goal check

return True

s = list(s) # turn sentence into list of charcters

symbol = s.pop(0) # remove the first character from the list

if m(symbol): # check the character for truth value in model

return PL-TRUE?(s,m) # if character passes recursively iterate through list

return False # return False if any characters fail

B. Consider a model that does not specify a truth value for apostrophes (‘). If the sentence does not have an apostrophe, then it can still be marked as true even with a partial model.

1. The quick lazy dog jumped over the brown startled fox.
2. The map on the wall charted the course of the great expedition with impressive detail.
3. The first explorers to set foot on Mars were surprised to discover the lack of oxygen.

C. If the model does not specify a truth value for all of the symbols, then each symbol in the sentence will need to be checked both for inclusion in the model and for a truth value. This would require adding an else if statement to the original algorithm to handle the case that nothing is returned from the model. If the model does not assign a truth value, then iteration would need to continue evaluating the rest of the sentence. Unlike the case where the model returns False, if the model does not evaluate the symbol, the rest of the symbols still need to be tested for a truth value to determine if the sentence is False or if the truth value of the sentence cannot be determined. This means the runtime of the algorithm is still linear with respect to the length of the sentence.

D.

Updated algorithm:

function PL-TRUE?(s, m) # s is the sentence and m is symbol truth model

if len(s) == 0: # goal check

return result

s = list(s) # turn sentence into list of charcters

symbol = s.pop(0) # remove the first character from the list

if m(symbol) == True: # check the character for truth value in model

result = True

return PL-TRUE?(s,m) # if character passes recursively iterate through list

elif m(symbol) == False:

return False # if one symbol fails, the entire sentence is False

else:

result = “undecided”

return PL-TRUE?(s,m) # if there is no truth value for symbol, continue with next

The algorithm will return True if all of the symbols in the sentence pass the truth assessment in the model, False if even a single one of the symbols is False, and will return “undecided” if one or more symbols were not evaluated by the model and the rest of the symbols were otherwise True.

Again, let us suppose that the model has no truth value for the apostrophe. The following sentences would return an “undecided” truth value:

1. The lazy dog jumped over the startled fox’s tail.
2. The map on the wall charted the explorer’s great expedition to unknown seas.
3. The first people on Mars were surprised to discover it wasn’t as much like an island paradise as they had been expecting.

E. Yes, this implementation of the algorithm makes TT-ENTAILS more efficient. The TT-ENTAILS algorithm passes a knowledge base (a sentence in propositional logic) and a model (to evaluate the truth of symbols) to PL-TRUE. If and only if the PL-TRUE algorithm returns True for the knowledge base does the algorithm move on to checking the statement α against the model. Therefore, if the knowledge base is not unambiguously, true, then α cannot be True and the TT-ENTAILS algorithm can stop and return False