1. Create a decider TM1 for the following language
   END_{DFA} = {<D, s> | D is a DFA and accepts at least one string that ends on symbol s}

   SOLUTION 1:
   The basic idea is to find all reachable states and examine incoming transitions – if at least one such transition in on symbol s then accepts otherwise reject.
   On input string <D,s>
   TM1 checks whether <D,s> is a valid encoding of a DFA and a symbol.
   if invalid then
       TM1 rejects <D,s>
   Else
       TM1 marks all reachable state of D using BFS
       For each marked state q do
             If q is a final state then
                 If q has an incoming transition t on symbols s then
                       TM1 accepts <D,s>
                 End if
             End if
       End for
       TM1 rejects <D,s>

   SOLUTION 2:
   The basic idea is to construct another DFA D1 that accepts all strings that ends on s, and then create a new DFA D2, which is the intersection of D and D1 and use E_{DFA} to determine whether they have some strings in common.
   On input string <D,s>
   TM1 checks whether <D,s> is a valid encoding of a DFA and a symbol.
   if invalid then
       TM1 rejects <D,s>
   Else
       TM1 generates a DFA D1 that accepts all strings that end on s
       TM1 create a DFA D2 such as L(D2) = L(D) ∩ L(D1)
       TM1 call the decider for DFA's Emptiness problem, E_{DFA}
       If EDFA(<D2>) accepts then
             TM1 rejects <D>
       Else
             TM1 accepts <D>
       End if
   End if

2. Create a decider TM2 for the following language
   $3SIZE_{DFA}$ = { <D> | D is a DFA and |L(D)| = 3}
   Recall that L(D) denotes the language of the machine D.

   SOLUTION:
   On input string <D>
   TM2 checks whether <D> is a valid encoding of a DFA
   if invalid then
        TM2 rejects <D>
   Else
        Create a counter count and sets it to 0, i.e., count = 0
        Create an empty stack S and pushed the start state of <D> on it
        TM2 performs the DFS using S with the following modifications:
                If a child of the top element of S is already on S then
                        TM2 rejects <D> (discovered a loop)
                Else
                        If a child of the top element of S is a final state then
                                Count = count + 1 (increments the counter)
                        End if
                End If
   If count == 3 then
        TM2 accepts <D>
   Else
        TM2 rejects <D>

3. Create a decider TM4 for the following language

   3StepsTM = {<T> | T is a TM and rejects the empty string within 3 steps, i.e., transitions}

SOLUTION:

On input string < T >

TM4 checks whether < T > is a valid encoding of a DFA and a string

If invalid then

       TM4 rejects < T >

Else

       For all i in {1, 2, 3} do

              TM4 simulates T on ε for i steps

              If T rejects then

                     M accepts < T >

              End if

       End for

       M rejects < T >

End if