1.

We can create a Turing Machine to compute g1(x) as follows:

T = “On input x, where x is <M> where M is a DFA:

1. If we fail the type check, output 0 and halt.
2. Simulate x on EDFA. If it accepts, output 0 and halt. Otherwise continue.
3. Evaluate *w*, where w is the smallest string (in the string order of Σ∗) in L(M).
4. Output 1*w*.

However, g2(x) does not have a Turing Machine that computes it.

Proof by contradiction: suppose there exists a Turing Machine that computes g2(x). Let us consider the cases where g2(x) will output 0. Our Turing Machine must take in <M> where M is a TM and L(M) = ∅. However, when computing if L(M) = ∅ is described by the set ETM, which is undecidable as well as unrecognizable. As we cannot construct a TM to compute ETM, it is a contradiction that our Turing Machine can compute g2(x).

2.

**a)**

A= Σ∗

B= HALTTMc

For any input x, x is within A and F(x) is within B. This is because Σ∗mapping reduces to every language other than ∅. We can also see from F that even if we do not have the correct type, it is outputted correctly as F(x) as if an incorrect type was passed as a complement of HALTTM, which is all the strings not in HALTTM.

**b)**

C=ATM

D=EQTM

In the case of x failing the type check, x is not in ATM, and we correctly output a string not in EQTM

In the case of x is not in ATM, we output the result of M’x with a TM that does not start with 0s, as is described by M’x . M’x rejects, and in step 4 we output two TMs that don’t have the same language, which is not in EQTM.

In the case of x is in ATM, then M’x accepts. We then output two TMs with the same language, which is in EQTM.

**c)**

X=HALTTM

Y=ETMc

In the case of x failing the type check, x is not in HALTTM, and we correctly output a string not in ETMc, which is an encoding of a machine in ETM.

In the case of x is not in HALTTM, we construct a TM with an empty language and output it, which is not in ETMc.

In the case of x is in HALTTM, we construct a TM without an empty language and output it, which is in ETMc.

3.

**a)**

Define F = “On input <M, w>, where M is a TM and w a string:

1. Run HALTTM on <M, w>. If it rejects, output <M, M, M>
2. Construct the Turing machine M’x = “On input y,

1. If M is a decider, accept.

2. Otherwise, reject.

1. Output <M’x>

We assume improperly formed inputs are assumed to map to strings outside of DECTM.

**b)**

Define F = “On input <M>, where M is a TM:

1. Construct the Turing machine M’x = “On input y,

1. If L(M) is an infinite set, accept.

2. Otherwise, reject.

1. Output <M’x>

We assume improperly formed inputs are assumed to map to strings outside of INFTM.

**c)** As we know ATM is undecidable, and proved ATM is mapping reducible to DECTM in part (a), it is evident that DECTM is undecidable.

**d)** As we know DECTM is undecidable from part (c), and proved DECTM is mapping reducible to INFTM in part (b), it is evident that DECTM is undecidable.