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 E_{DFA}. 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 E_{TM}, which is undecidable as well as unrecognizable. As we cannot construct a TM to compute E_{TM}, it is a contradiction that our Turing Machine can compute g2(x).

2.

a)

A= Σ^*

B= $HALT_{TM}^c$

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 $HALT_{TM}$, which is all the strings not in $HALT_{TM}$.

b)

C=A_{TM}

D=EQ_{TM}

In the case of x failing the type check, x is not in A_{TM}, and we correctly output a string not in EQ_{TM}

In the case of x is not in A_{TM}, 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 EQ_{TM}.

In the case of x is in A_{TM}, then M'_x accepts. We then output two TMs with the same language, which is in EQ_{TM}.

c)

$X = HALT_{TM}$

$Y = E_{TM}^c$

In the case of x failing the type check, x is not in $HALT_{TM}$, and we correctly output a string not in E_{TM}^c, which is an encoding of a machine in E_{TM}.

In the case of x is not in $HALT_{TM}$, we construct a TM with an empty language and output it, which is not in E_{TM}^c.

In the case of x is in $HALT_{TM}$, we construct a TM without an empty language and output it, which is in E_{TM}^c.

3.

a)

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

1. Run $HALT_{TM}$ 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.
3. Output <M'_x>

We assume improperly formed inputs are assumed to map to strings outside of DEC_{TM}.

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.
2. Output <M'_x>

We assume improperly formed inputs are assumed to map to strings outside of INF_{TM}.

c) As we know A_{TM} is undecidable, and proved A_{TM} is mapping reducible to DEC_{TM} in part (a), it is evident that DEC_{TM} is undecidable.

d) As we know DEC_{TM} is undecidable from part (c), and proved DEC_{TM} is mapping reducible to INF_{TM} in part (b), it is evident that DEC_{TM} is undecidable.