a) A_{ANY} = {<*M*> : TM *M* accepts at least one string}.

We show that A_{ANY} is not in D by reduction from H. Let *R* be a mapping reduction from H to A_{ANY} defined as follows:

R(<*M*, *w*>) =

1. Construct the description <*M#*> of a new Turing machine *M#*(*x*) that, on input *x*, operates as follows:
   1.1. Erase the tape.
   1.2. Write *w* on the tape.
   1.3. Run *M* on *w*.
   1.4. Accept.
2. Return <*M#*>.

If *Oracle* exists and decides A_{ANY}, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H. *R* can be implemented as a Turing machine. And *C* is correct. *M#* ignores its own input. It halts on everything or nothing. So:

- <*M*, *w*> ∈ H: *M* halts on *w*, so *M#* accepts everything. So it accepts at least one string. *Oracle*(<*M#*>) accepts.
- <*M*, *w*> ∉ H: *M* does not halt on *w*, so *M#* halts on nothing. So it does not accept even one string. *Oracle*(<*M#*>) rejects.

But no machine to decide H can exist, so neither does *Oracle*.

b) A_{ALL} = {<*M*> : = *L*(*M*) = Σ_M* }.

We show that A_{ALL} is not in D by reduction from H. Let *R* be a mapping reduction from H to A_{ANY} defined as follows:

R(<*M*, *w*>) =

1. Construct the description <*M#*> of a new Turing machine *M#*(*x*) that, on input *x*, operates as follows:
   1.1. Erase the tape.
   1.2. Write *w* on the tape.
   1.3. Run *M* on *w*.
   1.4. Accept.
2. Return <*M#*>.

If *Oracle* exists and decides A_{ALL}, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H. *R* can be implemented as a Turing machine. And *C* is correct. *M#* ignores its own input. It accepts everything or nothing. So:

- <*M*, *w*> ∈ H: *M* halts on *w*, so *M#* accepts everything. *Oracle* accepts.
- <*M*, *w*> ∉ H: *M* does not halt on *w*, so *M#* accepts on nothing. *Oracle* rejects.

But no machine to decide H can exist, so neither does *Oracle*.

c) {<*M*, *w*> : Turing machine *M* rejects *w*}.

We show that *L* is not in D by reduction from H. Let *R* be a mapping reduction from H to A_{ANY} defined as follows:

R(<*M*, *w*>) =

1. Construct the description <*M#*> of a new Turing machine *M#*(*x*) that, on input *x*, operates as follows:
   - 1.1. Erase the tape.
   - 1.2. Write *w* on the tape.
   - 1.3. Run *M* on *w*.
   - 1.4. Reject.
2. Return <*M#, w*>.

If *Oracle* exists and decides *L*, then *C* = *Oracle*(*R*(<*M*, *w*>)) decides H. *R* can be implemented as a Turing machine. And *C* is correct. *M#* ignores its own input. It halts on everything or nothing. So:

- <*M*, *w*> $\in$ H: *M* halts on *w*, so *M#* rejects everything. So, in particular, it rejects *w*. *Oracle* accepts.
- <*M*, *w*> $\notin$ H: *M* does not halt on *w*, so *M#* rejects nothing. So it does not reject *w*. *Oracle*(<*M#*>) rejects.

But no machine to decide H can exist, so neither does *Oracle*.