In programming languages, when a compiler compiles a program into machine language (or byte code, or any other intermediate form), it usually performs optimizations, to improve the speed or memory footprint of the code. For example, the compiler may attempt to remove unnecessary code, that will never be called. This next problem places a theoretical limit on that particular optimization.

A. Define $ENTER_{TM}$ to be the set of all strings $\langle M, q \rangle$, where M is a Turing machine, q is a state of M, and there exists a string w such that M enters q while processing w. Prove that $ENTER_{TM}$ is undecidable.

B. Problem 4.18. (Hint: Run the two Turing machines in parallel, by interleaving their steps. Then what?)

Recall that, for any two languages A and B, A/B is the set of all strings w such that there exists a string $x \in B$ such that $wx \in A$. In an earlier assignment, we showed that if A is regular, then A/B is regular. Our proof was non-constructive. This next problem suggests that the proof *must* be non-constructive. (I think that this is the last A/B problem in the course.)

C. Imagine a Turing machine N. It is designed to take input $\langle D, M \rangle$, where D is a DFA and M is a Turing machine. N processes this input $\langle D, M \rangle$, eventually halting with a string $\langle C \rangle$ on its tape, and nothing else. Here, C is a DFA such that $L(C) = L(D)/L(M)$. To be clear: N halts on all inputs, but we do not care whether it accepts or rejects them. We care about the contents of the tape upon halting. Prove that such an N cannot exist. (Hint: Use N to build a decider for $EMPTY_{TM}$.)