Theorem: For every NFA $N = (Q, \Sigma, \delta, s, F)$ we can build a DFA $M = (2^Q, \Sigma, \Delta, s, F)$ s.t. $L(M) = L(N)$. 

Key Defn.: A NFA $N$ can process a (sub-)string $y$ of length $n$ from state $p$ to state $q$ if there is a trace $T = (q_0, u_1, q_1, u_2, q_2, \ldots, q_{n-1}, u_n, q_n)$ such that: 
- for each $j$, $1 \leq j \leq n$, $(q_{j-1}, u_j, q_j) \in \delta$. (valid trace) 
- $q_0 = p$, $u_1 \cdot u_2 \cdots u_n = y$, and $q_n = q$. (Some $u_i$ can be $\epsilon$ later; other strings.

Given $p, r$, denote the set of such $y$ by $L_{pr}$. 

\[ L(N) = \bigcup_{q \in F} L_{qs} \]

Our proof will maintain the following Inductive Invariant as $i = 0$ for:

**INV.** The current state $r$ of the DFA equals the set of states $s \in Q$ s.t. $N$ can process $x_1 \cdots x_i$ from $s$ to $r$.

Proof: Take $S = \{ r : N$ can process $\epsilon$ from $s$ to $r \}$. (\( = \epsilon C q \) in text). This makes INV hold with $i = 0$ as the base case. Now let $i > 0$, assume INV hold for $i-1$ as the induction hypothesis. What will $i = n$ imply $L(M) = L(N)$?

- By INV for $i = N$, we want the final state $R_n$ of the DFA $M$ to equal the set of $r$ s.t. $N$ can process $x$ from $s$ to $r$.
- $R_n$ must be an accepting state exactly when some $q \in R_n$ belongs to $F$.

\[ S = R_n, \quad F = \{ r \in Q : R_n F \neq \emptyset \}. \]

Thus if we define $\Delta$ so that INV always holds, we will get $L(M) = L(N)$.

Note: We may assume $R_{i-1}$ comprises all states $s$ of $N$ can process $x_1 \cdots x_{i-1}$ from $s$ to $q$, including "all trailing $\epsilon$s": Text: $R_{i-1}$ is already $\epsilon$-closed.
Helpful auxiliary func: \( \Delta(p, c) = \{ r \mid N \text{ can process } c \text{ from } p \text{ to } r \} \)

Finally define, for all \( p \in Q \) and \( c \in \Sigma \):

\[
\Delta(p, c) = \bigcup_{p \in P} \Delta(p, c).
\]

To prove INV holds for \( x_1 \ldots x_i \), take \( C = x_i \) and instantiate \( P = \{ q_i \} \).

By Ind. Hyp., INV holds for \( x_{i-1} \). Let any \( r \in Q \) be given. We need: \( r \in \Delta(p) \).

First suppose \( r \in \{ q_i \} \), meaning \( N \) can process \( x_1 \ldots x_i \) from \( s \) to \( r \). Where was \( N \) after processing the last \( x_i \) before it did \( C = x_i \) ? It was in some state \( p \). By IH, \( p \in P \),

prime the nice \( x_1 \ldots x_{i-1} \). Hence the DFA \( M \) includes \( r \).

Conversely, suppose \( r \in \Delta(p, c) \) where \( c = x_i \) and \( P = \{ q_i \} \).

\[
r \in \bigcup_{p \in P} \Delta(p, c) \text{ so } r \in \Delta(p, c) \text{ for some } p \text{ in } \{ q_i \}.
\]

\[\vdots \]

\( N \) can process \( x_1 \ldots x_i \) from \( s \) to \( r \). Then process \( C = x_i \) and any trailing \( x \)s from \( r \) to \( r^* \). \( \because \) INV holds from \( 1 \) to \( n \) by induction.

---

Added: An example of tracing the states of an NFA. Repeated for convenience:

\[ X = \text{babaa} \]

Say: "All light bulbs are lit after two seconds."
Example for N. By the e-arc
write "Whenever 1 then also 3.
S = \{1, 2\} from before.
Use "Breadth-First Search"
(BFS)
\[s(1, a) = \{1\} \]
\[s(2, a) = \{2\} \]
\[s(1, 2) = \{1, 2\} \]
\[s(2, 1) = \{1, 2\} \]
\[\Delta(s) = s(1, a) \cup s(2, a) = \{1\} \cup \{2\} = \{1, 2\} \]
\[\Delta(s) = s(1, b) \cup s(2, b) = \{1\} \cup \{2\} = \{1, 2\} \]
\[\Delta(\{1, 2\}) = \{1, 2\} \cup \{1, 2\} = \{1, 2\} \]
\[\Delta(\{1, 2\}) = \{1, 2\} \cup \{1, 2\} = \{1, 2\} \]
\[\Delta(\{1, 2\}) = \{1, 2\} \cup \{1, 2\} = \{1, 2\} \]
\[\Delta(\{1, 2\}) = \{1, 2\} \cup \{1, 2\} = \{1, 2\} \]
\[\Delta(\{1, 2\}) = \{1, 2\} \cup \{1, 2\} = \{1, 2\} \]
\[\text{closed! Since 1,2\{ has already been seen, we say that }
\text{the BFS closed.}\]

Thus the DFA trace is the same as the "possible states trace" at each step of the NFA.
This is by design - it exemplifies the proof.

To answer the question of whether there is a string x, that "turns off all lightbubs" that
would be true if and only if M has a reachable dead state - M has no dead state, some
If M has a reachable dead state, it is when you get \(\emptyset\). Some other remarks:

- The states \(\{1\}\) and \(\{1, 2\}\) are impossible by the e-arc because they have 1 but not 2.
- The state \(\emptyset\) was possible but didn't come up in the "Breadth-First Search" - which here
  simply means "expanding" only those states you get from previous stages. Can you agree with
- Note that sometimes with Cartesian product you can "eliminate" because not all "pair" states overlap
  when the Boolean operation is \(\cup\) then it works the same as building \(s_1 \Delta s_2\) and then doing NFA \(N = \Delta s_1 s_2\) to DFA on \(N\)