Problem type 1:

L_1, L_2, ... are all regular languages representable by the DFAs $M_1 = (Q_1, \Sigma, \delta_1, s_1, A_1)$, $M_2 = (Q_2, \Sigma, \delta_2, s_2, A_2)$, etc.

Give the *formal* description of the NFA (N') that describes the language below that is the composite of one or more of the above regular languages.

(See variants below)

We want to see the definition in terms of: $Q' = ...$, $\delta' = ...$, $s' = ...$, $A' = ...$ Assume $\Sigma = \{\texttt{0}, \texttt{1}\}$.

a. **BYC**

$L' = L_1 \cup L_2$

b. **BYE**

L' = $\overline{L_1}$

c. **BYA**

$L' = L_1 \cdot L_2$

d. **BYH**

$L' = L_1 \cup \{\varepsilon\}$

(adding empty string to strings L_1 regardless of if it is there (or not))

e. **BYD**

$L' = \texttt{0} \cdot L_1$

Add a 0 before every string in L_1

f. **BYF**

$L' = L_1 \cup \{\texttt{0}\}$

(add the string "0" to strings in L_1 regardless of if it is there (or not))

g. **BYB**

$L' = L_1 \cdot \texttt{0}$

Add a 0 after every string in L_1

h. **BYG**

$L' = \{\texttt{0}^* x \texttt{0}^* a \texttt{0}^* y \texttt{0}^* | w = xay \in L_1, a \in \Sigma, x, y \in \Sigma^*\}$

(basically what this is saying is that L' will accept any string in L_1 and that string in L_1 can have any number of 0's shoved in between each character).