1. Recall the Bloom filter for approximate set membership: to store a set of \( n \) items, you create a bit array of length \( m \), and each item sets \( k \) random locations to 1. We showed in class that if \( k \approx \frac{m}{n} \ln 2 \) and \( m \approx \frac{1}{\ln 2} n \log_2(1/\delta) \) then this has no false negatives and a \( \delta \) chance of a false positive.

Now suppose there are \( \text{two} \) sets of size \( n \), \( A \) and \( B \), that are stored in Bloom filters with the same hash function \( h \). Let \( x_A \) and \( x_B \) be the corresponding bit arrays, and consider the bitwise AND of the two Bloom filters, \( y = x_A \& x_B \).

(a) Explain how to use \( y \) to estimate membership of \( A \cap B \). What are the false positive and false negative rates, in terms of \( n, m, k \), and \( |A \cap B| \)?

(b) Now optimize \( k \) to make the error rates as small as possible, for a fixed \( n \) and \( m \) (and for the worst case \( |A \cap B| \)). [Feel free to ignore issues of integrality and lower order terms.]

(c) What is the resulting \( m \) in terms of \( n \) and \( \delta \)? Compare this to regular Bloom filters.

(d) What is the expected number of 1s in \( x_A \) and \( y \), with the parameters you have produced? Compare this to the standard Bloom filter parameter setting.

(e) Go through the same argument for using the bitwise OR \( x_A \lor x_B \) to estimate \( A \cup B \).