Problem 1 (I/P, Folding)
Define the following functions using the higher-order function foldr from class:

a) and_list and or_list of type [Bool] -> Bool which return True if all (in the case of and_list) or at least one (in the case of or_list) of the list elements evaluates to True. Otherwise the functions should return False.

b) length :: [a] -> Int which computes the length (number of elements) of a list.

c) charcount :: Char -> [Char] -> Int which counts how many times a character occurs in a given character list. For example, charcount 's' "Haskell is fun" => 2

Hint: for some of these, it may be useful to define a separate “helper function” which is passed to foldr as the folding operation “o” (= first argument of foldr).

Problem 2 (I/P, Folding) The maximum (or minimum) of a list of numbers (say integer or floating point) cannot be directly defined using foldr since there is no suitable “start value” e (cf lecture notes). For such cases, we can use the functions myfoldl and myfoldr which have the signature (a -> a -> a) -> [a] -> a and which are defined as follows:

\[
\begin{align*}
\text{myfoldr} & \ (\oplus) \ [x_1, \ldots, x_n] \ = \ x_1 \oplus (x_2 \oplus (x_{n-1} \oplus x_n)) \\
\text{myfoldl} & \ (\oplus) \ [x_1, \ldots, x_n] \ = \ (x_1 \oplus (x_2 \oplus x_3) \ldots) \oplus x_n \\
\end{align*}
\]

a) Define myfoldr and myfoldl in Haskell.

b) Use either of those functions to define minlist and maxlist.