Exercise 4.72.

Why do `disjoin` and `stream-flatmap` interleave the streams rather than simply append them? Give examples that illustrate why interleaving works better. (Hint: Why did we use `interleave` in section 3.5.3?)

Answer.

`Disjoin` and `stream-flatmap` choose to interleave the streams instead of appending them so as to handle infinite streams properly (see section 3.5.3).

Once encountered infinite streams, the simple appending operation can lead to undesirable behavior. Suppose the first query Q_1 in an `or` combination extends an infinite input stream of frames S into another infinite one R_1, then the simple appending operation will take all the frames from R_1 before incorporating frames from $R_2, R_3, \ldots$, and hence will never produce any other frames stem from $R_2, R_3, \ldots$. Similarly, if the first stream of frames T_1 resulting from mapping a procedure over the input stream S is infinite, then all the rest stream of frames $T_2, T_3, \ldots$ will never get flatten if `stream-flatmap` accumulate the streams with an simple appending process.

However, using `interleave`, both `disjoin` and `stream-flatmap` will eventually reach every element they handle so long as permitted to run long enough. Since `interleave` takes elements alternately from the two streams, every element of the second stream will eventuallly find its way into the interleaved stream, even if the first stream is infinite.