Homework Assignment 3

Note 1: Solutions are expected to only use functions from the standard library that was taught. Before using a function from the standard library inquire if you are allowed to use it. If a solution uses a disallowed function, the autograder score is voided.
Note 2: only upload file `hw3.rkt`, as dependencies are available in server.
Note 3: Cannot use functions `append` and `reverse`.

1. Implement a function `(min-from n l)` that takes number `n` and a list `l` and returns the minimum number between `n` and every element of `l`. To calculate the minimum value between two values you can use `(min x y)`.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
2. Implement a function `(count l)` that takes a list `l` and returns the number of the elements in `l`.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
3. Implement a function `(sum l)` that takes a list `l` of numbers and returns the summation of every element of `l`.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
4. Implement a function `(occurrences x l)` that counts how many times `x` occurs in list `l`.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
5. Implement a function `(prefix s l)` that prepends string `s` before each string in `l`, in-order.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
   - Use `string-append` to prepend `s`.
6. Define function `(interleave l1 l2)` that returns a new list that interleaves each element of `l1` with each element of `l2`.
   - **Must be a recursive function.**
   - Must use the pattern-matching provided: one, and only one, `match` on `l1` with exactly 2 cases, a pattern for an empty list, and a pattern for a list with at least one element.
7. Implement a function `intersperse` that takes a list `l` and an element `e` and returns a list with the elements in list `l` interspersed with element `e`. That is, return a list where we add element `e` between each pair of elements in `l`.
   - Must be solved using either `foldr`, `foldl`, or `map`.
   - Definition cannot be recursive.
   - Try handling the case where the initial list is empty in one way and the case where the list is nonempty in another way.
   - This function is very similar to `join`.

8. Implement function `parse-ast` that takes a datum and yields an element of the AST.

   - You will have access to auxiliary functions `real?` and `symbol?` from Racket's standard library and functions `lambda?`, `define-basic?`, and `define-func?` from *your* file `hw1.rkt` from Homework Assignment 1 (Part II).
   - The function takes a datum that is a valid term.
   - Your function should only handle functions declarations, definitions, variables, and numbers.
   - Do **not** handle conditionals nor handle booleans.
   - You can request a partial solution of `hw1.rkt` if you forfeit the ability to resubmit Homework Assignment 1.