Question 1: Divisors

Write a function \( \text{divisors} \ n \) that consumes a non-zero \( \text{Nat} \) and returns a list containing the divisors of \( n \), in increasing order.

For example, the divisors of 12 are \( \{1, 2, 3, 4, 6, 12\} \), so \( \text{divisors} \ 12 \Rightarrow \text{list} \ 1 \ 2 \ 3 \ 4 \ 6 \ 12 \)

Question 2: Geometric Sequence

A geometric sequence is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed, non-zero number called the \textit{common ratio}. For example, the sequence 2, 6, 18, 54, \ldots is a geometric progression with common ratio 3.

Write a function \( \text{geo-seq} \ \text{start} \ \text{len} \ \text{ratio} \) that returns a \( \text{listof Num} \) containing the geometric sequence where the first value is \( \text{start} \), the length of the sequence is \( \text{len} \), and the common ratio is \( \text{ratio} \).

\( \text{geo-seq} \ 2 \ 4 \ 3 \Rightarrow \text{list} \ 2 \ 6 \ 18 \ 54 \)

Question 3: Currency Names

You are given a \( \text{listof (list Str Str)} \) containing pairs of country–currency, e.g.:

\begin{verbatim}
(define currencies
  (list (list "USA" "USD") (list "Norway" "NOK") (list "Switzerland" "CHF")
        (list "Japan" "JPY") (list "Canada" "CDN") (list "Germany" "EUR")))
\end{verbatim}

Write a function \( \text{lookup-currency} \ \text{country} \ \text{C} \) that consumes a \( \text{Str} \) and a \( \text{listof (list Str Str)} \). It returns the second \( \text{Str} \) of the item in \( \text{C} \) where the first item is \( \text{country} \).

For example, \( \text{lookup-currency} \ "Germany" \ \text{currencies} \Rightarrow "EUR" \)

\( \text{lookup-currency} \ "Canada" \ \text{currencies} \Rightarrow "CDN" \)

Note: you may assume that exactly one of the items in \( \text{C} \) has \( \text{country} \) as its first value. (Write this as a requirement.)

Use \texttt{filter}. What can you say about the list that \texttt{filter} returns?

Question 4: Bubble Sort

One way of sorting a list is to repeatedly swap adjacent out-of-order values in a list, until the list is in order. Each swap increases how sorted it is. After enough swaps, the list will be sorted.

For example, start at the right in \( \text{list} \ 11 \ 5 \ 13 \ 7 \).

- Compare the last two values, 13 and 7. 13 > 7, so swap the last two values, giving \( \text{list} \ 11 \ 5 \ 7 \ 13 \).
- Compare the second from last pair, 5 and 7. 5 < 7, so do nothing; still \( \text{list} \ 11 \ 5 \ 7 \ 13 \).
• Compare the third from last pair, 11 and 5. \(11 > 5\), so swap these values, giving \((\text{list } 5 \ 11 \ 7 \ 13)\).

By “bubbling” through the list once, \((\text{list } 11 \ 5 \ 13 \ 7)\) became \((\text{list } 5 \ 11 \ 7 \ 13)\), which is better sorted. Bubbling this list again gives \((\text{list } 5 \ 7 \ 11 \ 13)\), which is sorted.

Using \texttt{foldr}, write a function \texttt{bubble} that consumes a \((\text{listof Num})\), and returns the result of \textbf{one pass} of swapping out-of-order items in the list, \textbf{starting at the right}.

\begin{align*}
\text{(bubble (list 1 2 3 4))} & \Rightarrow \text{(list 1 2 3 4)} \\
\text{(bubble (list 11 5 13 7))} & \Rightarrow \text{(list 5 11 7 13)} \\
\text{(bubble (list 2 6 9 7 4 2 5 7))} & \Rightarrow \text{(list 2 2 6 9 7 4 5 7)}
\end{align*}

\textbf{Ex.} In the file that contains \texttt{bubble}, type in this program, and test it.

;;; (bsort L) return L, sorted in increasing order.
;;; bsort: (listof Num) -> (listof Num)
;;; Examples:
(check-expect (bsort (list 2 6 9 7 4 2 5 7))
  (list 2 2 4 5 6 7 7 9))

(define (bsort L)
  (foldr (lambda (a b) (bubble b))
    L
    (range 0 (length L) 1)))

\textbf{Ex.} Trace the code, and try to figure out why it works.