Symbolic differentiation using infix notation

You are provided a Scheme procedure and its helper procedures that perform symbolic
differentiation of polynomial expressions using **infix notation**:

\[
\frac{d}{dx} (ax^5 + bx^4 + 2x^3 + 6x^2 + 3x + 7) = 5ax^4 + 4bx^3 + 6x^2 + 12x + 3
\]

```
(deriv '(a x ^ 5 + b x ^ 4 + 2 x ^ 3 + 6 x ^ 2 + 3 x + 7) 'x)
⇒ (5 a x ^ 4 + 4 b x ^ 3 + 6 x ^ 2 + 12 x + 3)
```

Write additional procedures `evaluate` and `evaluate-deriv` to evaluate \( f(x) \) and \( f'(x) \),
respectively, for any value \( x \), where \( f \) is a polynomial function in \( x \) and \( f' \) is its derivative
with respect to \( x \). Assume a set of values for the coefficients \( a, b, c, \) etc. You can add
more helper procedures as needed.

Procedures `evaluate` and `evaluate-deriv` each must take two parameters, \( f \) and \( x \), where

- \( f \) is a list that represents the function’s polynomial expression in **infix notation**, an example of which is shown above.
- \( x \) is the value of \( x \) to evaluate \( f \) or \( f' \), respectively.

The coefficient values can be defined beforehand, such as with `(define a 1)`
Your procedures \texttt{evaluate} and \texttt{evaluate-deriv} must work at least for the example values below for $f$, $a$, and $b$, evaluated at $x$ bound to 0 and then to 1:

\begin{verbatim}
(define f '(a x ^ 5 + b x ^ 4 + 2 x ^ 3 + 6 x ^ 2 + 3 x + 7))
(deriv f 'x)  \Rightarrow  (5 a x ^ 4 + 4 b x ^ 3 + 6 x ^ 2 + 12 x + 3)

(define a 1)
(define b 1)

(evaluate f 0)  \Rightarrow  7
(evaluate f 1)  \Rightarrow  20

(evaluate-deriv f 0)  \Rightarrow  3
(evaluate-deriv f 1)  \Rightarrow  30
\end{verbatim}

You should, of course, test your procedures with other values.

\textbf{Tip}

Write helper procedures to convert the polynomial expression from infix to prefix notation.

\textbf{What to turn in}

Create a zip file containing:

- Text files containing your new Scheme procedures.
- Text files containing output from the above example values for $f$, $a$, $b$, and $x$, and other test values.
- A short report (a few paragraphs) that briefly explains your code design.

Email the zip file to ron.mak@sjsu.edu. Some mailers may not allow you to mail zip files, so you may have to rename the file to have the suffix other than .zip, such as .zzz. Do not include executable files.

\textbf{Important}: Name your zip file after your team name, such as \texttt{SuperCoders.zip}. Your email subject line should be: \texttt{CS 152 Assignment #3, team name} CC all your team members so I can “reply all” with your score.
Differentiate an infix polynomial with terms of the form \( +cax^n \)
where \( c \) is an optional integer constant and \( a \) is an optional variable other than \( x \), and optional exponent \( n \) is an integer > 0.
Example: \((3a x^5 + b x^4 + 2x^3 + 6x^2 + 3x + 7)\)
Differentiate with respect to \( x \), for example \( x \).
The derivative will be in the same infix form.

Assumptions:
The polynomial is in the "canonical" form as described above.
There is no error checking of the form. In particular, only
the variable we're differentiating with respect to (such as \( x \))
can have an exponent. Any deviation from canonical form may
cause a runtime exception.

Find the derivative of polynomial \( poly \) with respect to variable \( var \).
The polynomial must be in canonical infix form.
First "terminize" the polynomial.
Example:
\((\text{terminize } '(3a x^5 + b x^4 + 2x^3 + 6x^2 + 3x + 7))\)
\(\Rightarrow (\text{((3 a x^5) (b x^4) (2 x^3) (6 x^2) (3 x) (7)))}\)
Note that \( var \) is a free variable in local procedure \( \text{deriv-term} \).
At run time, \( \text{deriv-term} \) obtains the value of \( var \) from its closure,
and it is mapped over each sublist in the terminized polynomial.
The result is a list of sublists, each sublist representing the derivative
of the corresponding term.
Example:
\((\text{map deriv-term } '((3 a x^5) (b x^4) (2 x^3) (6 x^2) (3 x) (7))))\)
\(\Rightarrow (\text{((15 a x^4) (4 b x^3) (6 x^2) (12 x) (3) (0)))}\)
Example:
\((\text{deriv } '(3a x^5 + b x^4 + 2x^3 + 6x^2 + 3x + 7) \ 'x)\)
\(\Rightarrow (15 a x^4 + 4 b x^3 + 6 x^2 + 12 x + 3)\)

(define deriv
(lambda (poly var)
  (let* ((terms (terminize poly)) ; "terminize" the polynomial
        (deriv-term              ; local procedure deriv-term
          (lambda (term)
            (cond
              ((null? term) '())
              ((not (member? var term)) '0) ; deriv = 0
              ((not (member? ' ^ term)) (upto var term)) ; deriv = coeff
              (else (deriv-term-expo term var)) ; handle exponent
              ))))
    (diff (map deriv-term terms))) ; map deriv-term over the terms
    (remove-trailing-plus (polyize diff))) ; finalize the answer
))
Differentiate a single term that contains the var raised to a power (i.e., there's an exponent). If there are two adjacent integer constants in the new coefficient, replace them by their product.

Example: \((\text{deriv-term-expo } '(3 \ a \ x ^ 5) \ 'x) \Rightarrow (15 \ a \ x ^ 4)\)

(define deriv-term-expo
  (lambda (term var)
    (let* ((coeff (upto var term)) ; get the coefficient
           (rev-term (reverse term)) ; reverse so that we can
           (expo (car rev-term)) ; get the exponent at the end
           (new-expo (sub1 expo)) ; new exponent
           (new-coeff (if (number? (car coeff)) ; product
                        (cons (* (car coeff) expo) (cdr coeff))
                        (cons expo coeff))))
      (if (= new-expo 1)
          (append new-coeff (list var))
          (append new-coeff (list var '^ new-expo))))
    )))

; Convert an infix polynomial into a list of sublists, where each sublist is a term.
; Example: \((\text{terminize } '(3 \ a \ x ^ 5 + b \ x ^ 4 + 2 \ x ^ 3 + 6 \ x ^ 2 + 3 \ x + 7)) \Rightarrow ((3 \ a \ x ^ 5) (b \ x ^ 4) (2 \ x ^ 3) (6 \ x ^ 2) (3 \ x) (7)))\)

(define terminize
  (lambda (poly)
    (cond ((null? poly) '())
          (else (cons (upto '+ poly) (terminize (after '+ poly))))))
  ))

; Convert a list of term sublists to an infix polynomial.
; Do not include any + 0 term.
; There may be an extra + at the end.
; Example: \((\text{polyize } '((15 \ a \ x ^ 4) (4 \ b \ x ^ 3) (6 \ x ^ 2) (12 \ x) (3) (0))) \Rightarrow (15 \ a \ x ^ 4 + 4 \ b \ x ^ 3 + 6 \ x ^ 2 + 12 \ x + 3 +)\)

(define polyize
  (lambda (terms)
    (cond ((null? terms) '())
          ((equal? '(0) (car terms)) '())
          ((null? (cdr terms)) (car terms))
          (else (append (car terms) '(+) (polyize (cdr terms))))))
    )))
; Return the polynomial without any extra + at the end.
; If the polynomial is empty, return (0).
(define remove-trailing-plus
  (lambda (poly)
    (if (null? poly)
        '(0)
        (let ((rev-poly (reverse poly)))
          (if (equal? '+ (car rev-poly))
              (reverse (cdr rev-poly))
              (reverse rev-poly))
        ))))

; Return #t if the given item is a top-level item of the given list.
; Else return #f.
(define member?
  (lambda (item lst)
    (cond
      ((null? lst) #f)
      ((equal? item (car lst)) #t)
      (else (member? item (cdr lst))))))

; Return a list consisting of the items of the given list up to the first occurrence of the given item.
; Used to extract the coefficient of a term.
; Return the original list if the given item isn't in the list.
; Example: (upto 'x '(3 a x ^ 5)) ==> (3 a)
(define upto
  (lambda (item lst)
    (cond
      ((null? lst) '())
      ((equal? item (car lst)) '())
      (else (cons (car lst) (upto item (cdr lst)))))))

; Return a list consisting of the items of the given list after the first occurrence of the given item.
; Used to extract the exponent of a term.
; Return the empty list if the given item isn't in the list.
; Example: (after 'x '(3 a x ^ 5)) ==> (^ 5)
(define after
  (lambda (item lst)
    (cond
      ((null? lst) '())
      ((equal? item (car lst)) (cdr lst))
      (else (after item (cdr lst))))))