Use of procedure objects to make code more concise

Alternating harmonic series: \[ 1 - \frac{1}{2} + \frac{1}{3} - \cdots \pm \frac{1}{n} \]

Sign of last term depends on whether \( n \) is even or odd.

\[
\text{(define a-harm} \\
\text{(lambda (n))} \\
\text{(cond ((zero? n) 0)} \\
\text{((even? n) (- (a-harm (- n 1)) (/ 1 n)))} \\
\text{((odd? n) (+ (a-harm (- n 1)) (/ 1 n)))})\)

More concise:

\[
\text{(define a-harm} \\
\text{(lambda (n))} \\
\text{(if (zero? n) 0)} \\
\text{((if (even? n) - +)} \\
\text{\qquad (a-harm (- n 1)))} \\
\text{\qquad (/ 1 n))})\)

Syntactic sugar for *quote* special form

Can replace

\[(quote [SOMETHING])\]

with

\`[SOMETHING]\`

**Examples:**

Replace

\[(quote foo)\]

With

\`foo\`

Replace

\[(quote (a b c))\]

With

\`(a b c)`
Introduction to pattern matching

In pattern matching, we have a *pattern* and a *text*. We are looking for occurrences of the pattern in the text. What constitutes an occurrence depends on the problem.

Example problem: sublist?

Input: two lists of symbols, *pat* and *text*

Output: `#true` if the symbols in *pat* occur in *text*
- in same order as in *pat*
- consecutively

```lisp
(sublist? (quote (a b c)) (quote (x y a b c d)))  ; #true
(sublist? '(a b c) '(x y a b c d))  ; #true
(sublist? '(a b c) '(x y a b d c))  ; #false
```
Example problem: sublist?

**Input:** two lists of symbols, pat and text

**Output:** #true if the symbols in pat occur in text
  - in same order as in pat
  - consecutively

```
(sublist? (quote (a b c)) (quote (x y a b c d)))  \rightarrow #true

(sublist? '(a b c) '(x y a b c d))  \rightarrow #true

(sublist? '(a b c) '(x y a b d c))  \rightarrow #false
```
Example problem: sublist?
Input: two lists of symbols, \textit{pat} and \textit{text}
Output: \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}
\begin{itemize}
  \item in same order as in \textit{pat}
  \item consecutively
\end{itemize}

(sublist? (quote (a b c)) (quote (x y a b c d)))
(sublist? ‘(a b c) ‘(x y a b c d)) \rightarrow \texttt{#true}
(sublist? ‘(a b c) ‘(x y a b d c)) \rightarrow \texttt{#false}

Example problem: prefix?
Input: two lists of symbols, \textit{pat} and \textit{text}
Output: \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}
\begin{itemize}
  \item in same order as in \textit{pat}
  \item consecutively
  \item starting at beginning of \textit{text}
\end{itemize}

(prefix? ‘(c b a) ‘(c b a x y)) \rightarrow \texttt{#true}
Example problem: **prefix?**

**Input:** two lists of symbols, *pat* and *text*

**Output:** \texttt{#true} if the symbols in *pat* occur in *text*

- in same order as in *pat*
- consecutively
- starting at beginning of *text*

```
(prefix? '(c b a) '(c b a x y))  #true
```
Example problem: \texttt{prefix}?

\textbf{Input:} two lists of symbols, \textit{pat} and \textit{text}

\textbf{Output:} \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}

- in same order as in \textit{pat}
- consecutively
- starting at beginning of \textit{text}

\texttt{(prefix? `(c b a) `(c b a x y))} \rightarrow \texttt{#true}

\texttt{(define prefix? (lambda (pat text)
    (cond
      ((empty? pat) #true)
      (#true
        (if (equal? (car pat) (car text))
          (if (prefix? (cdr pat) (cdr text))
            #true
            #false)
          #false)))))}
Example problem: \texttt{prefix?}

Input: two lists of symbols, \textit{pat} and \textit{text}

Output: \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}

• in same order as in \textit{pat}
• consecutively
• starting at beginning of \textit{text}

\begin{verbatim}
(prefix? '(c b a) '(c b a x y)) → #true

(define prefix? (lambda (pat text) (cond ((empty? pat) #true) (#true (if (equal? (car pat) (car text)) (prefix? (cdr pat) (cdr text)) #false))))
\end{verbatim}
Example problem: prefix?

Input: two lists of symbols, \( \textit{pat} \) and \( \textit{text} \)
Output: \#true\text{ if the symbols in } \textit{pat} \text{ occur in } \textit{text}
\begin{itemize}
  \item in same order as in \textit{pat}
  \item consecutively
  \item starting at beginning of \textit{text}
\end{itemize}

\[
\text{(prefix? '(c b a) '(c b a x y))} \quad \rightarrow \quad \#\text{true}
\]

\[
\text{(define prefix?}
  \quad \text{(lambda (pat text)}
  \quad \text{  (cond}
  \quad \text{    ((empty? pat) #true)}
  \quad \text{    (#true}
  \quad \text{      (and (equal? (car pat) (car text))}
  \quad \text{        (prefix? (cdr pat) (cdr text))))))))
\]
Example problem: prefix?

Input: two lists of symbols, pat and text

Output: #true if the symbols in pat occur in text
  • in same order as in pat
  • consecutively
  • starting at beginning of text

(prefix? '(c b a) '(c b a x y)) → #true

(define prefix? (lambda (pat text)
  (or (empty? pat)
      (and (equal? (car pat) (car text))
           (prefix? (cdr pat) (cdr text))))))
Example problem: prefix?
Input: two lists of symbols, \textit{pat} and \textit{text}
Output: \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}
  • in same order as in \textit{pat}
  • consecutively
  • starting at beginning of \textit{text}

\texttt{(prefix? '(c b a) '(c b a x y))} \rightarrow \texttt{#true}

Example problem: sublist?
Input: two lists of symbols, \textit{pat} and \textit{text}
Output: \texttt{#true} if the symbols in \textit{pat} occur in \textit{text}
  • in same order as in \textit{pat}
  • consecutively

\texttt{(sublist? '(a b c) '(x y a b c d))} \rightarrow \texttt{#true}

Quiz: Write \texttt{sublist?} in terms of \texttt{prefix}?