Scheme Tutorial Solutions

Fall 2002

Problem Set 2: Lists and Trees

11. Functions which check to see is a temperature is between \(5^\circ C\) and \(95^\circ C\) inclusive.

\[
;\text{ between? : number \rightarrow boolean} \\
;\text{ checks if a number is between 5 and 95 inclusive} \\
(define \text{(between? x)}) \\
\text{(and \((\geq x\ 5)\ \(\leq x\ 95)))}
\]

\[
;\text{ check-range : (listof number) \rightarrow boolean} \\
;\text{ checks to see if each number in a list is between 5 and 95 inclusive} \\
(define \text{(check-range list-temps)}) \\
\text{(cond} \\
\text{[(empty? list-temps) true]} \\
\text{[(cons? list-temps) \text{(and (between? (first list-temps)) \text{(check-range (rest list-temps))}})])
\]

12. Functions which convert a list of digits to the corresponding number:

\[
;\text{ convert : (listof number) \rightarrow number} \\
;\text{ converts a list of digits to the corresponding number} \\
(define \text{(convert list-digits)}) \\
\text{(cond} \\
\text{[(empty? list-digits) 0]} \\
\text{[(cons? list-digits) \text{(+) \text{(first list-digits)} \text{(*) 10 \text{(convert (rest list-digits))}})}])
\]

13. Functions which average the values in a list

\[
;\text{ sum-prices : (listof number) \rightarrow number} \\
\text{(define \text{(sum-prices list-prices})} \\
\text{(cond} \\
\text{[(empty? list-prices) 0]} \\
\text{[(cons? list-prices) \text{(+ (first list-prices) \text{(sum-prices (rest list-prices))}})]})
\]
;; count-prices : (listof number) → number
(define (count-prices list-prices)
  (cond
   [(empty? list-prices) 0]
   [(cons? list-prices) (+ 1 (count-prices (rest list-prices)))]))

;; average-price : (listof number) → number
(define (average-price list-prices)
  (/ (sum-prices list-prices) (count-prices list-prices)))

14. Functions which convert a list of Celsius temperatures to Fahrenheit temperatures
;; f2c : number → number
;; converts a Fahrenheit temp to a Celsius temp
(define (f2c tempF)
  (* (/ 5 9) (– tempF 32)))

;; tempFC : (listof number) → (listof number)
;; converts a list of Fahrenheit temps to a list of Celsius temps
(define (tempFC list-Ftemps)
  (cond
   [(empty? list-Ftemps) empty]
   [(cons? list-Ftemps) (cons (f2c (first list-Ftemps))
                              (tempFC (rest list-Ftemps)))]))

15. Function which removes all items from a list which are greater than a user provided value
;; eliminate-exp : number (listof number) → (listof number)
(define (eliminate-exp ua lotp)
  (cond
   [(empty? lotp) empty]
   [(cons? lotp) (cond
                  [(< (first lotp) ua) (cons (first lotp)
                                            (eliminate-exp ua (rest lotp)))]
                  [else (eliminate-exp ua (rest lotp))])])

16. Function which consumes a list l and produces the list of all suffixes of l.
;; suffixes : (listof X) → (listof X)
(define (suffixes list-l)
  (cond
   [(empty? list-l) (cons empty empty)]
   [(cons? list-l) (cons list-l (suffixes (rest list-l)))]))
17. Datatype which represents a family tree

```scheme
(define-datatype family-tree family-tree?
  [unknown]
  [person (name string?)
    (birth-year number?)
    (eye-color symbol?)
    (mother family-tree?)
    (father family-tree?)])
```

18. Function which counts the number of people in a family-tree

```scheme
;; count-persons : family-tree number
;; counts the number of people in a family tree
(define (count-persons tree)
  (cases family-tree tree
    [unknown () 0]
    [person (name birth eye mom dad) (+ 1
      (count-persons mom)
      (count-persons dad))]))
```

19. Functions which computes the average age of the people in a family tree using the current year and their birth years.

```scheme
;; age-sum : family-tree number -> number
;; computes the sum of the ages of the people in a family tree
(define (age-sum tree year)
  (cases family-tree tree
    [unknown () 0]
    [person (name birth eye mom dad) (+ (- year birth)
      (age-sum mom year)
      (age-sum dad year))])

;; average-age : family-tree number -> number
;; computes the average age of the people in a family tree
(define (average-age tree year)
  (/ (age-sum tree year) (count-persons tree)))
```
20. Function which creates a list of the \textit{eye-color} of everyone in a \textit{family-tree}

\begin{verbatim}
;; eye-color : family-tree \rightarrow (listof symbol)
(define (eye-color tree)
  (cases family-tree tree
    [unknown () empty]
    [person (name birth eye mom dad) (cons eye
      (append (eye-color mom)
      (eye-color dad)))]))
\end{verbatim}