Problem Set 2: Lists and Trees

11. Functions which check to see if a temperature is between 5°C and 95°C inclusive.

;;; between? : number → boolean
;;; checks if a number is between 5 and 95 inclusive
(define (between? x)
  (and (>= x 5) (<= x 95)))

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

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

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

13. Functions which average the values in a list

;;; sum-prices : (listof number) → number
(define (sum-prices list-prices)
  (cond
   [(empty? list-prices) 0]
   [(cons? list-prices) (+ (first list-prices)
                          (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

\[
\text{(define-datatype family-tree family-tree?)}
\begin{align*}
& \text{[unknown]} \\
& \text{[person (name string?)}} \\
& \quad \text{(birth-year number?)}} \\
& \quad \text{(eye-color symbol?)}} \\
& \quad \text{(mother family-tree?)}} \\
& \quad \text{(father family-tree?)}}
\end{align*}
\]

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

\[
\text{;; count-persons : family-tree \rightarrow number}
\]
\[
\text{;; counts the number of people in a family tree}
\]
\[
\text{(define (count-persons tree)}
\]
\[
\text{(cases family-tree tree}
\]
\[
\quad \text{[unknown () 0]} \\
\quad \text{[person (name birth eye mom dad) (+ 1}
\]
\[
\quad \quad \quad \quad \text{(count-persons mom)}
\]
\[
\quad \quad \quad \quad \text{(count-persons dad))}})
\]

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

\[
\text{;; age-sum : family-tree number \rightarrow number}
\]
\[
\text{;; computes the sum of the ages of the people in a family tree}
\]
\[
\text{(define (age-sum tree year)}
\]
\[
\text{(cases family-tree tree}
\]
\[
\quad \text{[unknown () 0]} \\
\quad \text{[person (name birth eye mom dad) (+ (- year birth)}
\]
\[
\quad \quad \quad \quad \text{(age-sum mom year)}
\]
\[
\quad \quad \quad \quad \text{(age-sum dad year))}})
\]

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

```
;; eye-color : family-tree → (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)))]))
```