(define **toDec** ;deve convertire la stringa espressa con i simboli dell’insieme in numero decimale (lambda (insieme stringa) ;stringa = stringa di NUMERO INTERO insieme: valori in ordine crescente

(let ( (k (- (string-length stringa) 1)) ; K = ULTIMO INDICE DELLA STRINGA

(base (string-length base)) ;base = numero decimale da usare nei calcoli

)

(if (= k 0)

(nuovaCifra (string-ref stringa k) base 0) ;la stringa contiene un solo elemento

(+ ;NUOVACIFRA + BASE \* PREC

(\* base (toDec insieme (substring stringa 0 k))) ;funzione ricorsiva a cui viene passata la stringa senza l'ultimo carattere, che verrà sommato progressivamente

(nuovaCifra (string-ref stringa k) base 0) ) )))) ;viene calcolato il valore numerico del carattere all'ultima posizione della stringa

;restituisce le cifre prima del “.”

(define parte\_intera ;viene chiamata solo se la funzione

(lambda (stringa)

(let ((k (-(string-length stringa)1)))

(if (char=? (string-ref stringa k) #\.)

(substring stringa 0 k) ;restituisce la parte intera, perciò le cifre a SX della virgola

(parte\_intera (substring stringa 0 k)) )))) ;richiama se stessa togliendo l'ultimo carattere

(define parte\_fraz ;restituisce i caratteri a DX della virgola

(lambda (stringa)

(if (char=? (string-ref stringa 0) #\.)

(substring stringa 1)

(parte\_fraz (substring stringa 1)) ;richiama se stessa passando la stringa senza il carattere iniziale

)

))

;restuisce il valore numerico del carattere della stringa in base ai simboli dell'insieme. INDICE\_INSIEME = VALORE NUMERICO

(define nuovaCifra

(lambda (carattere insieme k) ;k è l'indice in cui cercare il carattere, vale 0 se è la prima volta che si chiama la funzione nuovaCifra

(if (< k (string-length insieme)) ;k deve essere minore della lunghezza della stringa, per non incorrere in una lettura fuori indice

(if (char=? (string-ref insieme k) carattere )

k ;CARATTERE TROVATO. Il valore numerico del carattere è rappresentato dall'indice

(nuovaCifra carattere insieme (+ k 1)) ;incrementa il contatore per cercare ancora nell'insieme

)

(string-append "ERRORE:" (string carattere) " non appartiene all'insieme") ;stampa a video dell'errore

)

)

)

(define checkSign ;restituisce il segno della stringa

(lambda (stringa)

(cond

((char=? (string-ref stringa 0) #\+)

#\+

)

((char=? (string-ref stringa 0)#\-)

#\-

)

(else

#\a ;non trova il segno perciò restituisce #\a come carattere indicante della mancanza del segno

)

)

)

)

(define hasPoint?

(lambda (string)

(let ((k (- (string-length string) 1)))

(cond

((char=? (string-ref string k) #\.) ;controlla se l'ultimo carattere è una virgola

#true

)

((= k 0)

#false ;siccome ha gia controllato se è presente il . allora ritorna falso

)

(else

(hasPoint? (substring string 0 k)) ;richiama se stessa togliendo l'ultimo carattere della stringa

)

)

)

)

)

(define bin-rep->number ;funzione chiamata dall'utente

(lambda (string)

(let ((insieme "01"))

(rep->number insieme string)

)

)

)

(define rep->number ;funzione chiamata dall'utente per conversione generica in BASE 10

(lambda (insieme stringa) ;insieme contiene i simboli dell'alfabeto da cui partire con la conversione

(if (hasPoint? stringa) ;controllo se il numero inserito ha la VIRGOLA

(let ( (numdec ;valore decimale della stringa totale senza la virgola

(toDec insieme (string-append

(parte\_intera (substring stringa 1))

(parte\_fraz (substring stringa 1))

)))

(divisore ;numero da dividere per ottenere la conversione giusta

(expt (string-length insieme) (string-length (parte\_fraz (substring stringa 1))) )

)

)

(cond ( (char=? (checkSign stringa) #\-)

(\* (/ numdec divisore) -1)

)

(else

;se è positivo o se non ha segno si fa la divisione normale

(/ numdec divisore)

)

)

)

;;NON TROVA LA VIRGOLA

(cond ( (char=? (checkSign stringa) #\+)

(toDec insieme (substring stringa 1) )

)

( (char=? (checkSign stringa) #\-)

(\* -1 (toDec insieme (substring stringa 1) ) )

)

(else

(toDec insieme stringa )

)

)

)

)

)

;ESERCIZIO 3 22-01-2018

;(powers-of-two 26) → (16 8 2)

(define powers-of-two ;restituisce la lista contenente le potenze di 2

;;in ordine decrescente che sommate danno origine a n

(lambda (n) ;n: INTERO >= 0

(cond ( (= n 0) ;CASO BASE

null

)

( (= n 1)

(list 1)

)

(else

(let ( (lg (inexact->exact

(floor (/ (log n) (log 2)) )

)

) ;Log2(n) : approssimato difetto INTERO

)

(cons (expt 2 lg) ;(potenza di 2) < n più vicina a n stesso

(powers-of-two (- n (expt 2 lg))) ; n - (potenza di 2)

)

)

)

)

)

)