(define **2D** ; valore: naturale

(lambda (down right) ; down, right: naturali

(if (or (= down 0) (= right 0))

1 ;se una delle 2 coordinate è 0, si ha 1 strada fino alla destinazione

(+ (2D (- down 1) right) ;procede verso il BASSO

(2D down (- right 1)) ;procede verso DESTRA )))) ;FINE

(define manhattan-3d

(lambda (down right k) ;k(3°dimensione), down, right: numeri interi

(cond ;se almeno 1 dimensione è 0 si passa alla valutazione BIDIMENSIONALE

((= down 0) (2D right k) )

((= right 0) (2D down k) )

((= k 0) (2D down right) )

(else (+ (manhattan-3d (- down 1) right k) ;procede verso il BASSO

(manhattan-3d down (- right 1) k) ;procede verso DESTRA

(manhattan-3d down right (- k 1)) ;procede verso la 3°dimensione )))));FINE

;ESERCIZIO 5 21-01-2019

;(mh 4 1) → (1 2 2 2 1) (mh 0 5) → (0)

(define mh ; val: lista di interi

(lambda (y x) ; x y interi che corrispondono alle coordinate del rispettivo ASSE

(if (or (= y 0) (= x 0)) (list 0)

(append (md (- y 1) x) ;si procede verso BASSE

(mr y (- x 1)) ;si procede verso DX ))))

;PASSO PRECEDENTE ASSE Y verso il BASSO

;ritorna i percorsi creati effettuando una mossa verso il BASSO

(define md ; md: passo precedente in giù,

(lambda (y x)

(cond ( (and (= y 0) (= x 0)) (list 0) ) ;PUNTO DI ARRIVO RAGGIUNTO

( (= y 0) (list 1) ) ;1 percorso disponibile, verso DX (x)

( (= x 0) (list 0) ) ;si può solo scendere

( else (append (md (- y 1) x) ;si procede verso il Basso e si prelevano le mosse successive

(map (lambda (x) (+ x 1) ) ;FUNZIONE: incrementa ogni valore della lista mr

(mr y (- x 1)) ) ))))) ;ARGOMENTO = LISTA si procede verso DX

;MOSSA PRECEDENTE ASSE X --> DX ;percorsi creati dopo effettuata una mossa verso DX

(define mr ; md: passo precedente a destra

(lambda (y x)

(cond ( (and (= y 0) (= x 0)) (list 0) ) ;PUNTO DI ARRIVO RAGGIUNTO

( (= y 0) (list 0) ) ;non si può più scendere

( (= x 0) (list 1) ) ;si deve scendere = 1 PERCORSO

(else (append (map (lambda (x) (+ x 1) ) ;FUNZIONE

(md (+ y 1) x) ) ;ARGOMENTO: LISTA

(mr y (- x 1)) ;si procede verso DX (x) )))));FINE

;ESERCIZIO 2 25-07-2019 ;LISTA DI TUTTI I PERCORSI MAHNATTAN

;(paths "12" "ab") → ("12ab" "1a2b" "1ab2" "a12b" "a1b2" "ab12")

(define paths

(lambda (u v)

(cond ( (string=? u "") (list v) ) ;rimane solo la dimensione v

( (string=? v "") (list u) ) ;rimane solo la dimensione u

(else (append

(map (lambda (s) (string-append (substring u 0 1) s )) (paths (substring u 1) v))

(map (lambda (s) (string-append (substring v 0 1) s )) (paths u (substring v 1))) )))))

;esercizio 4 22-01-2018

(define manhattan-var ; val: intero

(lambda (i j k) ; i, j, k: interi non negativi tali che k ≤ i e k ≤ j

(let ( (x (if (= i k) 0 (manhattan-var (- i 1) j k) ) )

(y (if (= j k) 0 (manhattan-var i (- j 1) k) ) )

(z (if (= k 0) 0 (manhattan-var (- i 1) (- j 1) (- k 1)) ) ) )

(if (and (> i 0) (> j 0))

(+ x y z )

1 )))) ;almeno una delle 2 coordinate = 0, perciò 1 solo percorso disponibile ma non in diagonale