(define posZ (char->integer #\Z) )

(define regola-cesare ;cambiando la procedura, cambia anche la chiave di crittografia e quindi l'interpretazione di un carattere

(lambda (key) ; chiave di crittazione passata durante la chiamata

(lambda (x) ;valore procedurale restituito dalla funzione ;x: CHAR

(integer->char

(let ( (i (+ key (char->integer x))) ) ;;cod ASCII di X shiftata di key

(if (<= i posZ) i (- i 26) ))))))

(define crittazione ;resituisce una stringa contente il msg cifrato in base alla funzione espressa da reg

(lambda (msg reg) ;msg: STRINGA , reg: funzione di crittazione

(if (string=? msg "") ""

(string-append (string (reg (string-ref msg 0)) ) ;coverte il primo carattere della stringa msg

(crittazione (substring msg 1) reg) ))))

;(crittazione (crittazione "CIAOCIAO" (regola-cesare 3)) testo cifrato

; (regola-cesare 23)

;) --> TESTO INIZIALE 23 + 3 = 26 = rotazione completa

;PROCEDURE CON ARGOMENTI E VALORI PROCEDURALI

(define posA (char->integer #\A))

(define regola-inversa ;restituisce la procedura decifr per decifrare

(lambda (cifra) ;cifr =procedura di crittazione

(let ( (k (char->integer (cifra #\A))) ) ;ASCII del char cifrato da A

(regola-cesare (- 26(- k posA))) ))) ;differenza tra il carattere cifrato e il carattere A

;corrisponde alla key di crittazione nella regola di cesare

;(define chiaveCrittazione (regola-cesare 3))

;(crittazione "CIAO" chiaveCrittazione) ;= "FLDR" -> testo shiftato di 3

;(crittazione "FLDR" (regola-inversa chiaveCrittazione) = (regola-cesare (26 - (k - posA))) ) = "CIAO"

(define regola-gen-inv ;valore: procedura inversa generale

(lambda (reg)

;bisogna scoprire il carattere inverso ;si usa una ricorsione finchè si trova la chiave giusta;BRUTE FORCE

(lambda (c) ;carattere da trovare

(trova reg c posA) ))) ;cod= codice ASCII del carattere progressivo da controllare

(define trova ;trova la chiave da restituire

(lambda (reg c i) ;indice = cod ASCII da controllare ricorsivamente ;c : carattere

(if (char=? (reg (integer->char i)) c)

(integer->char i) ;CHIAVE TROVATA

(trova reg c (+ i 1)) ))) ;ricerca successiva

;(define regola (regola-cesare 3))

;(crittazione "CIAO" regola) --> CIFRA IL TESTO CIAO = "FLDR"

;(crittazione "FLDR" (regola-gen-inv regola)) -->DECIFRA

;---------------------------------------------------------------------------------------------------

;;PARTE 1 es 9 laboratorio

;CIFRARIO DI CESARE DA MAIUSCOLO A MINUSCOLO ;numero da sommare a (char->integer #\X)

(define maiuscTominusc (- (char->integer #\a)(char->integer #\A)) )

(define alfabeto "ABCDEFGHILMNOPQRSTVX" ) ;20 CARATTERI

(define rotate ;shifta di key posizioni a DX il char secondo l'alfabeto

(lambda (i key char) ;i: contatore di ricerca in alfabeto

(cond ( (= i (string-length alfabeto))

char ) ;il contatore ha raggiuinto il limite e il carattere non è stato trovato

( (char=? (string-ref alfabeto i) char)

;i = posizione in cui è stato trovato il char

;CARATTERE TROVATO, viene restituito il carattere key posizioni a DX

(if (> (+ i key) (- (string-length alfabeto) 1))

;se la posizione da trovare è fuori range nell'alfabeto si torna indietro ai caratteri inizali

(string-ref alfabeto (- (+ i key) (string-length alfabeto)))

(string-ref alfabeto (+ i key)) ))

(else (rotate (+ i 1) key char) ))))

;restituisce la funzione di crittazione completa da applicare a un carattere

(define regola ;valore procedurale

(lambda (key) ;funzione di crittazione + MAIUSC -> MINUSC ;valore procedurale restituito dalla funzione

(lambda (c) ;c : CHAR MAISUCOLO da criptare

(integer->char (+ (char->integer (rotate 0 key c)) maiuscTominusc)) )))

(define rimuoviSpazi ;toglie i caratteri #\space da STRINGA

(lambda (s) ;s: STRINGA

(if (string=? s "")

s ;FINITI I CARATTERI

(if (char=? (string-ref s 0) #\space)

(rimuoviSpazi (substring s 1)) ;continua a togliere gli spazi

(string-append (substring s 0 1) ;primo carattere

(rimuoviSpazi (substring s 1)) )))))

;restituisce la FRASE (Stringa) inserita criptata grazie a reg

;reg = (regola key) 0<= key <= 19

;impostando key = 0 la funzione restituirà solo la conversione MAIUSC->minusc

;se key = 19 ogni carattere viene sostituito con il suo precendente nell'alfabeto

(define crittazione

(lambda (reg msg) ;reg: funzione di crittazione restituita dalla funzione regola

(let ( (listaMSG (string->list (rimuoviSpazi msg))) ;crea una lista di caratteri della stringa

)

(list->string (map reg

listaMSG )))))