;ES 8 02-12-2019

;(parity-check-failures '("0111" "1011" "0100" "1110")) → '(0 1 2 3)

;(parity-check-failures '("0110" "1101" "0000" "1011")) → '(1 3)

;QUALI ELEMENTI NON SUPERANO IL CONTROLLO

(define countBits ;conta i bit pari a "1"

(lambda (stringa)

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

(cond ( (= lungh -1) 0 ) ;CASO BASE stringa vuota

( (char=? (string-ref stringa lungh) #\1)

(+ 1 (countBits (substring stringa 0 lungh) ) ));toglie l'ultimo carattere

(else ;l'ultimo carattere = 0

(countBits (substring stringa 0 lungh) ) )))))

(define parity-check-failures ;ritorna le stringhe con un numero DISPARI di "1"

(lambda (lista) ;lista di stringhe binarie

;controlla le stringhe dall'ultima alla prima, salvando la posizione come eventuale ritorno

(let ((lungh (- (length lista) 1)) )

(cond ( (null? lista) null ) ;lista vuota CASO BASE

( (= (remainder (countBits (car (reverse lista))) 2) 0) ;l'ultimo elemento contiene un numero PARI di 1

(parity-check-failures (reverse (cdr (reverse lista))) )) ;toglie l'ultimo elemento e ricerca

;non SUPERA il controllo

(else (append (parity-check-failures (reverse (cdr (reverse lista))) ) ;toglie l'ultimo elemento

(list lungh) )))))) ;posizione dell'elemento che non supera il controllo

;ESERCIZIO 2 13-07-2021

(define parity-check ; val : stringa binaria

(lambda (w) ; w : lista non vuota di stringhe binarie della stessa lunghezza

(parity-rec (cdr w) (car w)) ))

(define match ;restituisce la stringa con i valori di parità

(lambda (s p)

(if (string=? s "") "";STRINGA VUOTA

(if (char=? (string-ref s 0) (string-ref p 0))

;i primi due caratteri sono uguali perciò si aggiunge 0

(string-append "0" ;si aggiunge 1

(match (substring s 1) (substring p 1))) ;senza il primo carattere

;DIVERSI si aggiunge 1

(string-append "1"

(match (substring s 1) (substring p 1)) ))))) ;senza il primo carattere

(define parity-rec ; val : stringa binaria

(lambda (w p) ; w : lista di stringhe binarie, p : prima stringa binaria della lista precedente

(if (null? w) p

(parity-rec (cdr w) (match (car w) p)))))