Correction Projet n° 1 Classe de Seconde (Michel LAGOUGE)

```
;codage de Cesar
(require (lib "trace.ss"))

;programme principal
; avec saisie du texte à coder ou décoder

(define (principal)
  (display "Programme  Codage<=> Decodage de Cesar")
  (newline)
  (display "choix possibles :")(newline)
  (display "1 : Encodage") (newline)
  (display "2 : Decodage avec la clé connue") (newline)
  (display "3 : Decodage avec la clé inconnue")(newline)
  (let ((rep (begin (display "votre choix : ") (read))))
    (cond ((= 1 rep)
          (let((text (begin (display "entrez votre texte(entouré de
guillemets !) : -fin par retour chariot-) ")(read)))
               (decal (begin (display "entrez votre clé (décalage donc un
entier dans intervalle [0..25]): ")(read))))
           (if (or (not (number? decal)) (< decal 0) (> decal 25))
               (error "Entrée incorrecte pour la clé!" decal)
               (encode_Cesar (mise_en_forme text) decal))))
          ((= 2 rep)
          (let((text (begin (display "entrez votre texte codé(entouré de
guillemets !) : -fin par retour chariot-) ")(read)))
               (decal (begin (display "entrez votre clé (décalage donc un
entier dans intervalle [0..25]): ")(read))))
           (if (or (not (number? decal)) (< decal 0) (> decal 25))
               (error "Entrée incorrecte pour la clé!" decal)
               (decode_Cesar (mise_en_forme text) decal))))
          ((= 3 rep)
          (let((text (begin (display "entrez votre texte codé(entouré de
guillemets !) : -fin par retour chariot-) ")(read))))

               (decode_Cesar_total (mise_en_forme text) )))
          (else (error "Entrée incorrecte!" rep)))))

;utiliser la fonction suivante pour tout mettre en minuscule (string-
downcase un_string)
; rappel des codes  #\a <=> 97   #\z<=>122  #\espace<=>32


;version simple
(define (encode_Cesar str decalage);  str est un string entièrement en
minuscule et pas de caractere accentué
  (define (encode_char code_car)
    ; ne sont décalés que les caractères correspondant à des lettres, les
autres sont inchangés
    (if (or (< code_car 97)(> code_car 122))
        code_car
        (let ((val (+ code_car decalage)))
          (if (> val 122) (- val 26) val))))
    (list->string (map integer->char (map  encode_char (map char->integer
(string->list str))))))
```

```
;> (encode_Cesar  "ceci est un essai !" 2)
;"egek guv wp guuck !"
;> (encode_Cesar  "ceci est un essai !" 24)
;"acag cqr sl cqqyg !"
;(encode_Cesar  "les eleves de l'ecole alsacienne sont tres bons en
informatique !" 7)
;"slz lslclz kl s'ljvsl hszhjpluul zvua aylz ivuz lu pumvythapxbl !"


(define (decode_Cesar str decalage);  str est un string entièrement en
minuscule
  (define (decode_char code_car)
    ; ne sont décalés que les caractères correspondant à des lettres, les
autres sont inchangés
    (if (or (< code_car 97)(> code_car 122))
        code_car
        (let ((val (- code_car decalage)))
          (if (< val 97) (+ val 26) val))))
    (list->string (map integer->char (map  decode_char (map char->integer
(string->list str))))))

;> (encode_Cesar "abcd efg!" 2)
;"cdef ghi!"
;> (decode_Cesar(encode_Cesar "abcd efg!" 2) 2)
;"abcd efg!"
;(encode_Cesar "ceci est un essai !" 1)
;"dfdj ftu vo fttbj !"
;> (decode_Cesar(encode_Cesar "ceci est un essai !" 1) 1)
;"ceci est un essai !"
;> (encode_Cesar "zorro est arrive !" 5)
;"etwwt jxy fwwnaj !"
;> (decode_Cesar(encode_Cesar "zorro est arrive !" 5) 5)
;"zorro est arrive !"
;-------------------------------------------------------------------------
----------------------
; on suppose maintenant que le décalage pour le décodage n'est pas connu
; il faut donc faire rechercher les fréquences de chaque caractère
; et supposer que le #\e a la plus grande frequence   .. ce qui permet de
retrouver le décalage

; Les fréquences sont dans des A_listes ( car freq)
; construction de la A liste
(define (constr_liste n decal) ; construit une liste (n  n+ 1..... n+ decal)
  (define (travail acc index) ; la fonction qui travaille pour avoir une
récursivité terminale
      (if (zero? index)
          (cons n acc)
          (travail (cons (+ n index) acc) (- index 1))))
  (travail () decal))

;> (constr_liste 97 25)
;(97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115
116 117 118 119 120 121 122)


(define (constr_Alist liste); construit la liste de frequence vide    Alist
:  ((car freq)...)
  (map (lambda (nb) (list (integer->char nb) 0)) liste))
; exemple
; (constr_Alist '(97 98 99 100))
;((#\a 0) (#\b 0) (#\c 0) (#\d 0))
```

```
 (define (constr_Alist_ex liste); construit la  Alist : ((car
code_Ascii)...)
  (map (lambda (nb) (list (integer->char nb) nb)) liste))
;exemple
; (constr_Alist_ex (constr_liste 97 7))
;((#\a 97) (#\b 98) (#\c 99) (#\d 100) (#\e 101) (#\f 102) (#\g 103) (#\h
104))
; suite projet

(define (constr_Alist_code liste); construit la liste de frequence vide
Alist :  ((ca_code freq)...)
  (map (lambda (nb) (list nb 0)) liste))

(define (constr_freq Str_list) ; construit la liste des frequences argument
une liste de caractères
 (define (travail Str_list A_list)
    (if (null? Str_list)
        A_list
        (travail (cdr Str_list)
(map (lambda(assoce) (if (char=? (car Str_list) (car assoce ))
                         (list (car assoce ) (+ 1 (cadr assoce)))
                         assoce))
     A_list))))
  (travail Str_list (constr_Alist (constr_liste 97 25))))
; exemple
;(constr_freq (string->list "ceci est un essai !"))
;((#\a 1) (#\b 0) (#\c 2) (#\d 0)(#\e 3)(#\f 0)(#\g 0)(#\h 0)(#\i 2)(#\j 0)
;(#\k 0)(#\l 0)(#\m 0)(#\n 1)(#\o 0)(#\p 0)(#\q 0)(#\r 0)(#\s 3)(#\t 1)
;(#\u 1)(#\v 0)(#\w 0)(#\x 0)(#\y 0)(#\z 0))

(define (rech_car_freq_max A_list) ;
; retourne une liste correspondant aux caracteres -il peut y en avoir
plusieurs correspondant à la frequence max
 (define (travail A_list max list_car)
   (if (null? A_list)
       ;(list list_car max) ; retourne la liste de caracteres et la
frequence max
       list_car  ; retourne uniquement la liste de caracteres de frequence
max
       (travail (cdr A_list)
                (if (> (cadar A_list) max) (cadar A_list) max)
                (if (> (cadar A_list) max) (list (caar  A_list))
                    (if (= (cadar A_list) max) (cons (caar  A_list)
list_car) list_car)))))
  (travail A_list 0 ()))
; exemple
; (rech_car_freq_max (constr_freq (string->list "ceci est un essai !")))
;(#\s #\e) nb_occurrence max = 3 pour #\s et #\e

(define (calcule-decal car)
  (let ((decalage (- (char->integer car) (char->integer #\e))))
    (if (negative? decalage)
        (+ decalage 26)
        decalage)))

(define (decode_Cesar_total str);  str est un string entièrement en
minuscule, le decalage est inconnu !
  ; recherche des décalages possibles
  (let ((list_decal (map calcule-decal(rech_car_freq_max (constr_freq
(string->list str))))))
    (map (lambda (x) (decode_Cesar str x)) list_decal)))
```