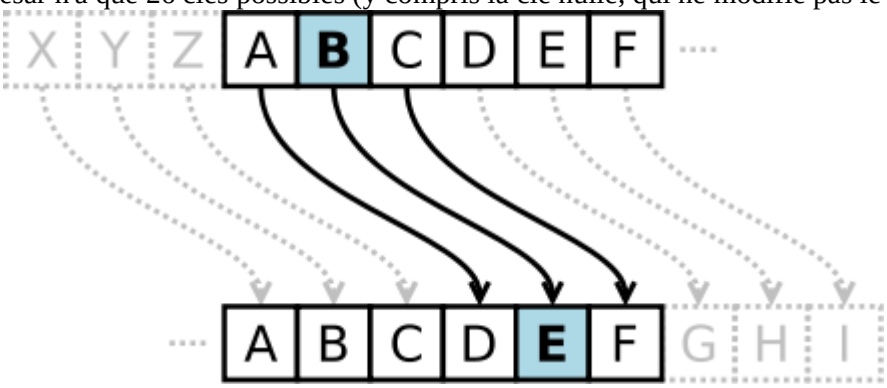
Projet n° 1 Classe de Seconde (Michel LAGOUGE)
Programmation DrRacket

Préliminaire

L'actualité montre que lors du transfert d'information, il est important de coder les données afin que l'information ne soit pas accessible à n'importe qui. C'est l'objet de la ***cryptographie***.
On se propose dans ce projet de mettre en œuvre un codage très simple appelé ***codage par décalage .*** Ce codage est encore appelé « ***codage de César*** » car utilisé par Jules César dans ses correspondances secrètes.

Principe du « codage de César »

Le texte chiffré s'obtient en remplaçant chaque lettre du texte clair original par une lettre à distance fixe, toujours du même côté, dans l'ordre de l'alphabet. Pour les dernières lettres (dans le cas d'un décalage à droite), on reprend au début. Par exemple avec un décalage de 3 vers la droite, A est remplacé par D, B devient E, et ainsi jusqu'à W qui devient Z, puis X devient A etc. Il s'agit d'une permutation circulaire de l'alphabet. La longueur du décalage, 3 dans l'exemple évoqué, constitue la clé du chiffrement qu'il suffit de transmettre au destinataire — s'il sait déjà qu'il s'agit d'un chiffrement de César — pour que celui-ci puisse déchiffrer le message. Dans le cas de l'alphabet latin, le chiffre de César n'a que 26 clés possibles (y compris la clé nulle, qui ne modifie pas le texte).



Exemple avec un décalage (clé) de 3

Soit la correspondance entre les ***alphabets***

```
clair   : ABCDEFGHIJKLMNOPQRSTUVWXYZ
chiffré : DEFGHIJKLMNOPQRSTUVWXYZABC
```

Les seules informations que doivent partager l'expéditeur et le destinataire du message sont :
- Le message codé
- La clé du codage

Evidemment ces deux informations ne doivent pas être ensemble afin de ne pas permettre à n'importe qui de déchiffrer le message codé.

Exemple :
Message en clair : "les eleves de l'ecole alsacienne sont tres bons en informatique ! "
Message codé avec une clé de 7 : "slz lslclz kl s'ljvsl hszhjpluul zvua aylz ivuz lu pumvythapxbl ! "

Problèmes pour un message en français :
- Les lettres avec accents
- La ponctuation
- Les espaces
- Les majuscules

On se propose dans un premier temps de ne considérer que des messages très simples :
- Pas de majuscules
- Pas de caractères avec accent
- … par contre, on garde les espaces et la ponctuation (*exactement comme dans l'exemple ci-dessus*)

I Première partie du projet

Ecrire les deux fonctions suivantes : (encode_Cesar *str* *decalage*) :

- *arguments :*

str : une chaîne de caractères
decalage : un entier correspondant au décalage (la clé)

- *retour :*

une chaîne de caractères codée

(decode_Cesar *str* *decalage*) :

- *arguments :*

str : une chaîne de caractères ne comportant que des caractères minuscules, des espaces et des caractéres de ponctuation
decalage : un entier correspondant au décalage (la clé)

- *retour :*

une chaîne de caractères décodée

a) Version simple

Les chaînes de caractères en argument ne comportent que des caractères minuscules sans accents , des espaces et des caractères de ponctuation.
La chaîne retournée par les deux fonctions ne comportent que des caractères minuscules sans accents , des espaces et des caractères de ponctuation.

b) Version améliorée

Les chaînes de caractères en argument comportent des caractères avec majuscules, minuscules, accents , des espaces et des caractères de ponctuation
La chaîne retournée par les deux fonctions ne comportent que des caractères minuscules sans accents , des espaces et des caractères de ponctuation.

II Deuxième partie du projet : « casser le code »

On se place maintenant du point du vue du hacker qui veut « casser le code » et bien sûr, n'a pas la clé du codage. Le hacker sait qu'en français, la lettre la plus fréquente est le 'e'.

Par conséquent, sur le message codé, il va chercher la fréquence de chaque lettre dans le message codé, rechercher la lettre qui a la fréquence maximum (*mais attention, plusieurs lettres peuvent avoir une fréquence maximum !)* et admettant que cette (ou ces lettres !) correspondent à 'e', déterminer la clé pour pouvoir, enfin, décoder le message.

Ecrire la fonction (decode_Cesar_total *str*)

- *arguments :*

str : une chaîne de caractères ne comportant que des caractères minuscules sans accent , des espaces et des caractères de ponctuation.

- *retour :*

une liste de chaînes de caractères décodée (éventuellement ne comportant qu'une seule chaîne) ne comportant que des caractères minuscules sans accent , des espaces et des caractères de ponctuation.

Exemples :

```
> (decode_Cesar_total "acag cqr sl cqqyg !")
("oqou qef gz qeemu !" "ceci est un essai !")
```

; une liste de deux chaînes de caractères car dans le message de départ, le 'e' et le 's' ont la même frequence maximale = 3

```
> (decode_Cesar_total "slz lslclz kl s'ljvsl hszhjpluul zvua aylz ivuz lu
pumvythapxbl !")
("les eleves de l'ecole alsacienne sont tres bons en informatique !")
```

; ici une liste ne comportant qu'une seule chaîne puisque le 'e' a la fréquence maximale et pas d'autre lettre.

Des informations utiles…

Ne pas hésiter pour compléter les informations données ci-dessous à utiliser l'aide de DRacket

A. *Le caractère ASCII associé à chaque lettre*

A chaque caractère du clavier est associé un code ASCII qui est un entier.
Voir http://www.toutimages.com/codes_caracteres.htm

Pour ***la version simple***, il suffit de savoir que les lettres minuscules allant de 'a' à 'z' correspondent à un code ASCII allant de 97 à 122. Ils sont accessibles par la fonction car->integer.

Exemples :

```
> (char->integer #\a)
97
> (char->integer #\z)      ; noter comment s'écrit un caractères   #\....
122
```

Pour ***la version amelioree***, il faudra transformer la lettre accentuée dans la lettre correspondante non accentuée et pour cela, récupérer les codes ASCII de toutes les lettres accentuées en français (é,è, ë, ù, ü, etc.)

Exemple:

```
> (char->integer #\é)
233
```

http://www.toutimages.com/codes_caracteres.htm pour récupérer les codes des autres lettres

B. Les chaînes de caractères

En DrRacket, les chaînes de caractères (String en anglais et en informatique » s'écrivent avec des guillemets

On peut transformer tous les caractères d'un String soit tout en minuscules, soit tout en majuscules :

```
> (string-upcase "abc!")
"ABC!"
> (string-upcase "Straße")
"STRASSE"
```

```
> (string-downcase "aBC!")
"abc!"
> (string-downcase "Straße")
"straße"
```

C. Chaîne de caractères et listes

On peut transformer un String en liste de caractères :

```
> (string->list "abc!")
(#\a   #\b   #\c   #\!)
```

Et inversement… une liste de caractères en String

```
> (list->string '(#\a #\b #\c #\!))
"abc!"
```

D. Liste d'Associations

Une A_list ou liste d'Associations est une liste composée de sous listes associant deux données.
Exemple : A-list associant un caractère minuscule et son code ASCII
((#\a 97) (#\b 98) (#\c 99) (#\d 100) (#\e 101) (#\f 102) (#\g 103) (#\h 104)….)

Cette structure peut être intéressante pour constituer la liste des fréquences assciées àchaque caractère d'un String.

E. La fonction map

Cette fonction permet d'executer la même fonction sur tous les éléments d'une liste.

Exemples

```
> (define (1+ n) (+ 1 n))
> (define (carre n) (* n n))
> (define liste '( 1 2 3 4 5))
> (map 1+ '( 1 2 3 4 5))
(2 3 4 5 6)
> (map carre liste)
(1 4 9 16 25)
>(define list_nb '(97 98 99 100 101 102 103 104 105 106))
> (map (lambda (nb) (list (integer->char nb) nb)) list_nb)
((#\a 97) (#\b 98) (#\c 99) (#\d 100) (#\e 101) (#\f 102) (#\g 103) (#\h
104) (#\i 105) (#\j 106))
```

```
; Soit le programme principal
; avec saisie du texte à coder ou décoder
; la fonction mise_en_forme se charge de transformer le texte saisie en
éliminant les majuscules, les caractères accentués pour ne garder que des
caractères minuscules… et la ponctuation

(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)))))
```

Exemples d'exécution :

```
> (principal)
Programme  Codage<=> Decodage de Cesar
choix possibles :
1 : Encodage
2 : Decodage avec la clé connue
3 : Decodage avec la clé inconnue
votre choix : 3
entrez votre texte codé(entouré de guillemets !) : -fin par retour chariot-)
"acag cqr sl cqqyg !"
("oqou qef gz qeemu !" "ceci est un essai !")

>(principal)
Programme  Codage<=> Decodage de Cesar
choix possibles :
1 : Encodage
2 : Decodage avec la clé connue
3 : Decodage avec la clé inconnue
votre choix : 5
```

 Entrée incorrecte! 5 ***; Il s'agit d'un message d'erreur !***