```
;Representaremos alfabetos como Strings.
;Por ejemplo, si nuestros simbolos son las cinco primeras letras,
los digitos y el espacio,
;lo representaremos como "ABCDE0123456789"

;Representaremos simbolos como strings de longitud 1. En el
alfabeto anterior,
;el simbolo E lo representamos con el string "E"

;El codigo del Cesar lo representaremos mediante parejas de
simbolos.
;Por ejemplo, si queremos decir que el simbolo "A" se codifica con
el
;simbolo "C", tendremos (make-Tupla "A" "C") para representar esta
situacion.

;Primero comenzamos definiendo algunas funciones sobre strings y
listas que nos son de utilidad.

;partir : String -> List(String)
;Dado un string, devuele una lista de strings con cada simbolo
separado.

(check-expect (partir "ABC") (list "A" "B" "C"))
(check-expect (partir "12345") (list "1" "2" "3" "4" "5"))
(check-expect (partir "") empty)

(define (partir s)
  (map string (string->list s)))

;tomar : List(Natural) Natural -> List(Natural)
;Dada una lista y un numero natural n, devuelve una lista con los
primeros n elementos de l.
;Si l no tiene tantos elementos, devuelve l.

(check-expect (tomar (list 1 2 3 4 5) 4) (list 1 2 3 4))
(check-expect (tomar (list 1 2 3 4 5) 10) (list 1 2 3 4 5))
(check-expect (tomar (list 1 2 3 4 5) 0) empty)
(check-expect (tomar empty 5) empty)

(define (tomar l n)
  (cond [(or (empty? l) (= n 0)) empty]
        [else (cons (first l) (tomar (rest l) (sub1 n)))]))

;tirar : List(Natural) Natural -> List(Natural)
;Dada una lista y un numero natural n, devuelve una lista sin los
primeros n elementos de l.
;Si l no tiene tantos elementos, devuelve empty.

(check-expect (tirar (list 1 2 3 4 5) 2) (list 3 4 5))
```

```
(check-expect (tirar (list 1 2 3 4 5) 10) empty)
(check-expect (tirar (list 1 2 3 4 5) 0) (list 1 2 3 4 5))
(check-expect (tirar empty 3) empty)

(define (tirar l n)
  (cond [(or (empty? l) (= n 0)) l]
        [else (tirar (rest l) (sub1 n))]))

(define-struct Tupla [f s])
;Tupla es [Any Any]
;que representa un par de elementos de cualquier tipo.

;emparejar : List(X) List(Y) -> List(Tuplas)
;Dadas dos listas [a0,..., an] y [b0, ...., bn] de la misma longitud, devuelve una lista
;de tuplas con parejas tomadas de ambas listas: [(make-posn a0 b0), ...., (make-posn an bn)]

(check-expect (emparejar (list "a" 2) (list "b" 4)) (list (make-Tupla "a" "b") (make-Tupla 2 4)))
(check-expect (emparejar (list "h" "l") (list "o" "a")) (list (make-Tupla "h" "o") (make-Tupla "l" "a")))
(check-error (emparejar '() (list "o" "a")) "emparejar : espera lista no vacia.")
(check-error (emparejar (list "h" "l") '()) "emparejar : espera lista no vacia.")
(check-error (emparejar (list "h" "l") (list "h" "l" "p")) "emparejar : espera dos listas con la misma longitud.")

(define (emparejar l1 l2)
  (cond
    [(and (empty? l1) (empty? l2)) empty]
    [(empty? l1) (error "emparejar : espera lista no vacia.")]
    [(empty? l2) (error "emparejar : espera lista no vacia.")]
    [(not (= (length l1) (length l2))) (error "emparejar : espera dos listas con la misma longitud.")]
    [else (cons (make-Tupla (first l1) (first l2)) (emparejar (rest l1) (rest l2)))]))

;CIFRADO DEL CESAR

;cifrado : Natural String -> List(Tupla)
;Dada una clave de desplazamiento y un alfabeto s, devuelve una lista
;con parejas de strings, donde el primer elemento es el caracter a cifrar, y el segundo
;su codigo del Cesar de acuerdo a la clave. Se asume que 0 < n < (string-length s).

(check-expect (cifrado 2 "ABC") (list (make-Tupla "A" "C")
```

```
(make-Tupla "B" "A") (make-Tupla "C" "B")))
(check-expect (cifrado 1 "ABC") (list (make-Tupla "A" "B")
(make-Tupla "B" "C") (make-Tupla "C" "A")))

(define (cifrado clave s)
  (emparejar (partir s) (append (tirar (partir s) clave) (tomar
(partir s) clave))))

;encriptar-simbolo : String List(Tupla) -> String
;Dado un string s de longitud 1 que es un simbolo del
;alfabeto y una lista de parejas que representa un codigo del
Cesar,
;devuelve el codigo que le corresponde a s.

(check-expect (encriptar-simbolo "A" (cifrado 2 "ABC")) "C")
(check-expect (encriptar-simbolo "A" (cifrado 1 "ABC")) "B")

(define (encriptar-simbolo s l)
  (cond [(empty? l) #f]
        [(string=? s (Tupla-f (first l))) (Tupla-s (first l))]
        [else (encriptar-simbolo s (rest l))]))

;encriptar-mensaje : String String Natural -> String
;Dado un string, un alfabeto y una clave, devuelve el string
encriptado.

(check-expect (encriptar-mensaje "ABC" "ABCDEF" 3) "DEF")
(check-expect (encriptar-mensaje "ABC" "ABCDEF" 4) "EFA")

(define (encriptar-mensaje s alfabeto clave)
  (local [  ;encriptar-simbolo-aux : String -> String
          (define (encriptar-simbolo-aux str)
            (encriptar-simbolo str (cifrado clave alfabeto)))]
    (foldr string-append "" (map encriptar-simbolo-aux (partir
s)))))

;desencriptar-simbolo : String List(Tupla) -> String
;Dado un string s de longitud 1 que es un simbolo del
;alfabeto y una lista de parejas que representa un codigo del
Cesar,
;devuelve el caracter desencriptado que le corresponde a s.

(check-expect (desencriptar-simbolo "A" (cifrado 2 "ABC")) "B")
(check-expect (desencriptar-simbolo "A" (cifrado 1 "ABC")) "C")

(define (desencriptar-simbolo s l)
  (cond [(empty? l) #f]
        [(string=? s (Tupla-s (first l))) (Tupla-f (first l))]
        [else (desencriptar-simbolo s (rest l))]))
```

```
;desencriptar-mensaje : String String Natural -> String
;Dado un string, un alfabeto y una clave, devuelve el string encriptado.

(check-expect (desencriptar-mensaje "DEF" "ABCDEF" 3) "ABC")
(check-expect (desencriptar-mensaje "EFA" "ABCDEF" 4) "ABC")

(define (desencriptar-mensaje s alfabeto clave)
  (local [ ;desencriptar-simbolo-aux : String -> String
           (define (desencriptar-simbolo-aux str)
             (desencriptar-simbolo str (cifrado clave alfabeto)))]
    (foldr string-append "" (map desencriptar-simbolo-aux (partir s)))))

;EVALUACION DE EXPRESIONES

(define ALFABETO "ABCDEFGHIJKLMNÑOPQRSTUVWXYZ 0123456789")
(define CLAVE 3)

(encriptar-mensaje "HOLA" ALFABETO CLAVE)
(encriptar-mensaje "ATACAR A LAS 18" ALFABETO CLAVE)
(encriptar-mensaje "LA OPERACION ES REVERSIBLE" ALFABETO CLAVE)
(desencriptar-mensaje (encriptar-mensaje "LA OPERACION ES REVERSIBLE" ALFABETO CLAVE) ALFABETO CLAVE)
```