009. Buscar y reemplazar un elemento en una lista

Dados dos elementos X e Y y una lista, obtener otra lista en la cual todas las ocurrencias de X hayan sido reemplazadas por Y.

`(reemplaza-elemento buscado reemplazante lst)`

donde cada ocurrencia de **`buscado`** en **`lst`** será reemplazada por **`reemplazante`** en la lista de salida.

Veamos unos ejemplos:

```
(reemplaza-elemento ‘a ‘b ‘(c d e)) -> (c d e) # no existe el buscado
(reemplaza-elemento ‘a ‘b ‘(b c d)) -> (b c d) # no existe el buscado
(reemplaza-elemento ‘a ‘b ‘(1 2 a b c)) -> (1 2 b b c) # una vez
(reemplaza-elemento ‘a ‘b ‘(1 a c a)) -> (1 b c b) # dos veces
(reemplaza-elemento ‘a ‘b ‘()) -> () # reemplazos en la lista vacía
```

Analicemos los casos posible que tenemos cuando miramos la cabeza de la lista:

- la cabeza es el buscado => agregar el reemplazante a la salida
- la cabeza no es el buscado => agregar la cabeza a la salida
- la lista está vacía => el resultado es la lista vacía

En los dos primeros casos, se debe continuar con los reemplazos en la cola de la lista, para obtener la salida.

```
(define reemplaza-elemento
  (lambda (buscado reemplazante lst)
    (cond ((null? lst) ‘())
          ((equal? (car lst) buscado)
           (cons reemplazante
                 (reemplaza-elemento buscado reemplazante (cdr lst))))
          (else (cons (car lst)
                 (reemplaza-elemento buscado reemplazante (cdr lst)))) )))
```

Debemos controlar si está vacía la lista antes de intentar buscar la cola o cabeza, las cuales no están definidas para la lista vacía.