Escribe el cuerpo de la siguiente función que dada una pareja, comprueba si su parte izquierda es una cadena de caracteres y su parte derecha un símbolo. Utiliza las funciones **string?** y **symbol?**

```
(define (pareja-cadena-simbolo? pareja)
    
)
```

```
(check-true (pareja-cadena-simbolo? '("hola" . lpp)))
(check-false (pareja-cadena-simbolo? '(hola . hola)))
(check-false (pareja-cadena-simbolo? '("lpp" . "lpp")))
```

Suponiendo definidas las funciones **(mayor x y)** y **(menor x y)** que devuelven el mayor y el menor de los dos parámetros, completa la definición de la función **(expt-mayor-menor x y)** que devuelve el mayor número elevado al menor.

```
(define (expt-mayor-menor x y)
    (expt (mayor x y) (menor x y))
)
```

```
(check-equal? (expt-mayor-menor 2 3) 9)
(check-equal? (expt-mayor-menor 5 2) 25)
(check-equal? (expt-mayor-menor 3 10) 1000)
```

Completa el código de la función **(simbolo-ref->lista s ref)** que devuelve una lista con el carácter del símbolo que está en la posición indicada por ref.

```
(define (simbolo-ref->lista s ref)
    (list (string-ref (symbol->string s) ref))
)

(check-equal (simbolo-ref->lista 'ABCD 2) '(#\C))
```

Dadas las siguientes funciones:

```
(define (f x y)
   (+ (* 2 x) (/ y 2))

(define (g x)
   (if (>= x 0) x (* x -1)))
```

Escribe la expresión resultante de aplicar el siguiente paso del modelo de sustitución usando el orden aplicativo:

```
(f 3 (g 2))
```

Respuesta: (f 3 (if (>= 2 0) 2 (* 2 -1)))

```
(define (expt-mayor-menor x y)
  (expt (mayor x y) (menor x y)))


(define (foo n)
  (integer->char (- (char->integer #\9) n)))

(foo 5) ; ->  #\4
```

Dada el siguiente código en Scheme:

```
(define (f x)
   (+ x 2))

(f (f 4))
```

Reescribe completa la expresión anterior aplicando el primer paso del modelo de sustitución utilizando el orden normal:

Respuesta: (+ (f 4) 2) ✔