Examen mañana

Ejercicio 2

a)

```
(cons (cons 5
            (cons 7
                  (cons 8 7)))
      (cons (cons 2 '())
            (cons 1 (cons (cons 2 7)
                    '()))))
```

b) Sí es una lista, porque cumple la definición de lista: una pareja donde su parte izquierda contiene el primer elemento de la lista y la parte derecha el resto de la lista. Si observamos el resto de la lista hasta llegar al final de la lista, llegamos a la última pareja donde su parte izquierda contiene el último elemento (la pareja (2.7)) y su parte derecha el resto de la lista (fin de lista), lista vacía.

Ejercicio 3

```
(define (cumple-predicados lista n)
    (if (null? lista)
        '()
        (cons ((car lista) n)
              (cumple-predicados (cdr lista) n))))
```

Ejercicio 4

a) ((g suma-1) 2)
b) 9

Ejercicio 5

a)

```
(define (n-primeros lista n)
   (if (= n 0)
      '()
      (cons (car lista) (n-primeros (cdr lista) (- n 1)))))

(define (n-ultimos lista n)
    (if (= (length lista) n)
        lista
        (n-ultimos (cdr lista) n)))
```

b)

```
(define (mover-n lista n)
   (if (null? lista)
       '()
       (append (n-ultimos lista n) (n-primeros lista (- (length lista) n)))))
```

Ejercicio 6

a)

```
(define (construye-poli lista1 lista2)
    (if (null? lista1)
        '()
        (cons (cons (car lista1) (car lista2))
              (construye-poli (cdr lista1) (cdr lista2)))))
```

b)

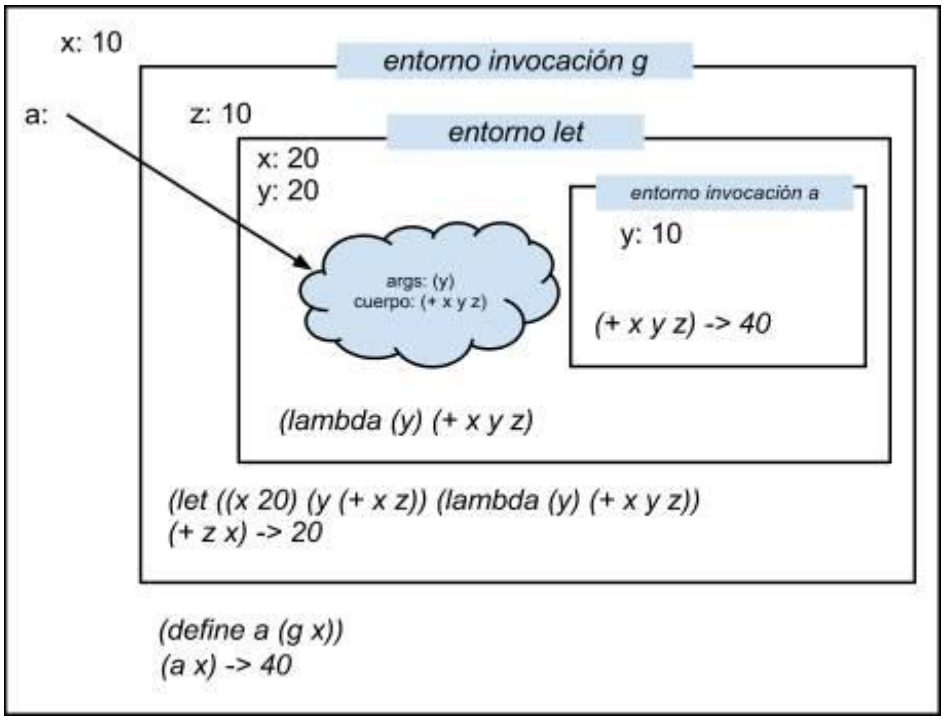
```
(define (coords-min-x poli)
    (if (null? (cdr poli))
        (car (car poli))
        (let ((min (coords-min-x (cdr poli))))
          (if (> (car (car poli)) min)
              min
              (car (car poli))))))

(define (coords-min-y poli)
    (if (null? (cdr poli))
        (cdr (car poli))
        (let ((min (coords-min-y (cdr poli))))
          (if (> (cdr (car poli)) min)
              min
              (cdr (car poli))))))

(define (coords-min poli)
   (cons (coords-min-x poli) (coords-min-y poli)))
```

Ejercicio 7

a) Devuelve 40
b)



c) Sí que se ha creado una clousure, la función generada por la expresión lambda y guardada en la variable a. La función se crea en el ámbito del let y se ejecuta en ese mismo ámbito, usando las variables allí definidas.

Examen tarde

Ejercicio 1

b) HTML no es un lenguaje de programación. Tiene sintaxis pero no es ejecutable. Una página HTML define el aspecto de una página web. Es leída por un navegador, que pinta la página tal y como se indica en la especificación HTML. Pero no contiene instrucciones a realizar por un ordenador. El lenguaje HTML no contiene primitivas ni reglas de combinación y abstracción de esas primitivas.

Ejercicio 2

a)

```
(cons (cons 5
            (cons 7
                  (cons 8 7)))
      (cons (cons 2 '())
            (cons 1
                  (cons (cons 2 '())
                        2))))
```

b) No es una lista, porque no cumple la definición de lista: una pareja donde su parte izquierda contiene el primer elemento de la lista y la parte derecha el resto de la lista. Si observamos el resto de la lista hasta llegar al final de la lista, llegamos a la última pareja donde su parte izquierda contiene el último elemento (la lista (2)) y su parte derecha no contiene el resto de la lista (fin de lista), sino que contiene el dato atómico 2.

Ejercicio 3

```
(define (resultados lista n)
    (if (null? lista)
        '()
        (cons ((car lista) n)
              (resultados (cdr lista) n))))
```

Ejercicio 4

a)

```
(define f
  (lambda ()
      (lambda (x)
          (+ x x))))
((f) 5) → 10
```

b)

```
(define (g x)
     (+ x 1))

(define (f g)
    (lambda (x)
        (g x)))

((f g) 3) → 4
```

Ejercicio 5

a)

```
(define (primeros-n lista n)
     (if (= n 0)
         '()
         (cons (car lista) (primeros-n (cdr lista) (- n 1)))))

(define (ultimos-n lista n)
    (if (= (length lista) n)
        lista
        (ultimos-n (cdr lista) n)))
```

b)

```
(define (intercalar lista)
  (intercalar-aux
     (primeros-n lista (/ (length lista) 2))
     (ultimos-n lista (/ (length lista) 2))))

(define (intercalar-aux lista1 lista2)
  (cond ((null? lista1) lista2)
        ((null? lista2) lista1)
        (else (cons (car lista1) (intercalar-aux lista2 (cdr lista1))))))
```

Ejercicio 6

a)

```
(define (genera-parejas a b simbolo)
    (if (> a b)
        '()
        (cons (cons a simbolo)
              (genera-parejas (+ a 1) b simbolo))))

(define (construye-baraja-aux lista-palos)
   (if (null? lista-palos)
      '()
      (append
         (genera-parejas 1 12 (car lista-palos))
         (construye-baraja-aux (cdr lista-palos)))))

(define (construye-baraja)
   (construye-baraja-aux '(oros copas bastos espadas)))
```

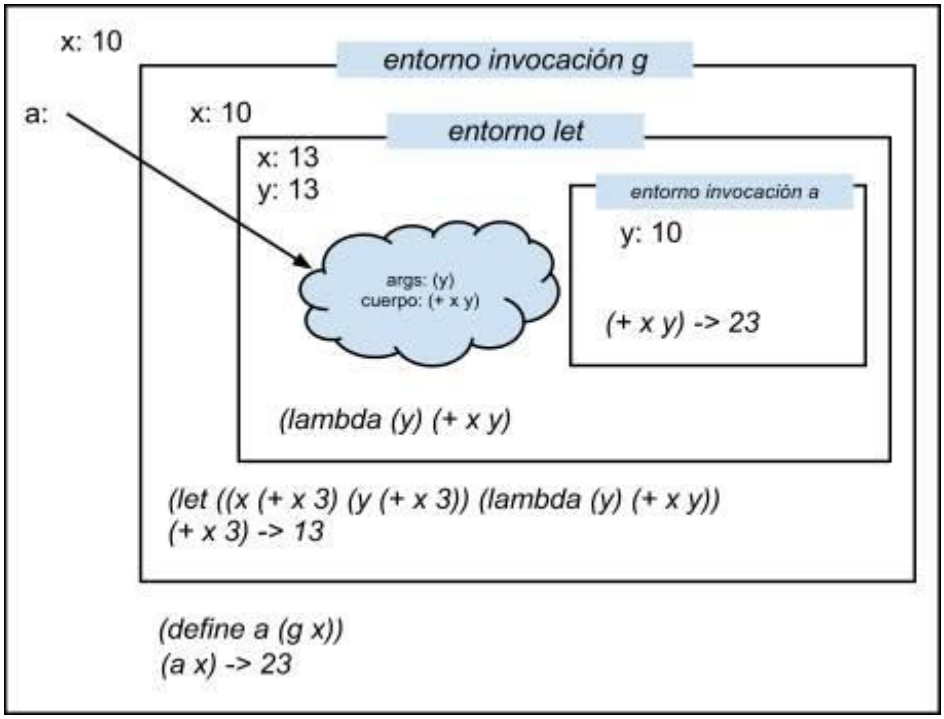
b)

```
(define (total-palo palo mano)
   (if (null? mano)
       0
       (if (equal? (cdar mano)) palo)
            (+ (cddr mano) (total-palo palo (cdr mano)))
            (total-palo palo (cdr mano)))))
```

Ejercicio 7

a) 23

b)



c) Sí que se ha creado una clousure, la función generada por la expresión lambda y guardada en la variable a. La función se crea en el ámbito del let y se ejecuta en ese mismo ámbito, usando las variables allí definidas.