Ejercicios de programación funcional

Se recuerda que:

- En las funciones recursivas deberá realizarse previamente el análisis por casos utilizando el esquema dado en las sesiones de prácticas.
- Siempre que sea posible, la definición de funciones de orden superior (FOS) se han de *currificar*. Proporcionar siempre una versión sin *currificar* y, posteriormente, la versión *currificada*.

1. Definir la función recursiva de orden superior $add(x, l, f)$ que dado un número x, y una lista l de números ordenados según el criterio establecido por la función de comparación f, retorna una nueva lista añadiendo x en la posición de l que le corresponda según el criterio de comparación dado.

   Dados dos elementos cualesquiera a y b de la lista l, la función de comparación $f(a, b)$ retornará cierto si el elemento a se encuentra antes que b en l y falso en caso contrario.

   Ejemplo: `(add 5 '(10 8 7 3 1) >) => (10 8 7 5 3 1)`

2. Utilizar la función previa (add) para definir la función $add\text{-}length(l, ll)$ que añade la lista l a una lista ll cuyos elementos son listas y están ordenados según su longitud en sentido creciente.

   Ejemplo: `(add-length '(a b) '((a) (1) (x y (z)))) => ((a) (1) (a b) (x y (z)))`

3. Dar la definición recursiva de la función $addLast(x, l_0, l_1, \ldots l_{n-1})$ que dado un elemento x y una o más listas li retorne la lista $(l'_0\ l'_1 \cdots l'_{n-1})$, donde cada l'_i es la misma lista li con el elemento x añadido al final ($0 \leq i < n$).

   Ejemplo: `(addLast '(1) '(a b) '((d) e f) '()) => ((a b (1)) ((d) e f (1)) ((1)))`

4. Proporcionar la función $addLast{+}(x, l_0, l_1, \ldots l_{n-1})$ redefiniendo la función del apartado previo ($addLast$) para validar sus argumentos. En el caso de que éstos sean incorrectos deberá proporcionarse el mensaje de error correspondiente.

5. Dada la función recursiva $enteros$ que se define al final, tal que $enteros(a, b)$ retorna la lista de enteros consecutivos $(a\ \ a+1\ \ a+2\ \ \cdots\ \ b-2\ \ b-1)$ y que estará vacía si $a \geq b$, utilizar ésta y FOS para definir la función $suc0(r, n)$. Esta función ha de retornar la lista $(0\ \ r\ \ 2*r\ \cdots\ \ (n-1)*r)$; es decir, la lista de los n primeros términos de la sucesión aritmética de razón r y cuyo primer elemento es 0.

```
(define (enteros x y)
  (if (>= x y)
      ()
      (cons x (enteros (+ x 1) y)))
```

```
(enteros -2 4) => (-2 -1 0 1 2 3)
(enteros 1 6) => (1 2 3 4 5)
```

   Ejemplo: `(suc0 5 7) => (0 5 10 15 20 25 30)`

6. Utilizar la función $suc0$ del ejercicio previo y FOS para definir la función $sucesion(x, r, n)$ que retorna la lista de los n primeros términos de la sucesión aritmética de razón r y cuyo primer elemento es x. Define también la función $serie(x, r, n)$ que retorna la suma de los elementos de la sucesión.

   Ejemplo: `(sucesion -5 3 6) => (-5 -2 1 4 7 10)` y `(serie -5 3 6) => 15`

```
(define UNO
  '((ana    ((4 rojo) (9 rojo) (9 verde) (comodin-color) (9 verde) (8 verde) (3 azul)))
    (rosa   ((0 amarillo) (1 amarillo) (3 rojo) (8 azul) (4 verde) (3 rojo) (9 azul)))
    (luis   ((comodin-color) (8 amarillo) (5 rojo) (7 amarillo) (5 verde) (4 amarillo)))
    (pedro  ((7 amarillo) (1 rojo) (4 verde) (9 rojo) (7 rojo) (6 amarillo) (4 rojo)))
    (maria  ((7 verde) (8 amarillo) (0 rojo) (comodin-roba4) (3 verde) (8 verde) (3 azul)))
    (carmen ((8 azul) (5 verde) (5 amarillo) (6 amarillo) (3 amarillo) (6 azul) (roba2 verde)))
    (blanca ((3 amarillo) (1 rojo) (5 amarillo) (4 amarillo) (2 azul) (9 azul) (7 azul)))
    (quique ((9 amarillo) (reversa rojo) (2 rojo) (6 verde) (8 rojo) (1 azul) (1 verde)))))
```

7. En el párrafo previo se define el símbolo UNO que contiene la información de los jugadores de una partida de cartas del UNO, por cada jugador la lista:

   (*nombre-del-jugador* (*mano-del-jugador*))

   Si las cartas del UNO se representan mediante la lista (*valor color*), exceptuando los comodines que sólo tienen valor, y los colores y valores que pueden tomar las cartas son los siguientes:

   - Los colores $rojo$, $verde$, $azul$ y $amarillo$
   - Los valores numéricos enteros del 0 al 9 y los valores de las *cartas de acción*, los símbolos: $roba2$, $reversa$, $salta$ y los comodines de color $comodin\text{-}color$ y $comodin\text{-}roba4$

   Se pide:

   7.1. Definir mediante FOS la función $jugador(nombre, juego)$ que dado el nombre de un jugador de la partida de UNO retorna su información

   Ejemplo:
   ```
   (jugador 'rosa UNO)
          => (rosa ((0 amarillo) (1 amarillo) (3 rojo)
                    (8 azul) (4 verde) (3 rojo) (9 azul)))
   ```

   7.2. Definir mediante FOS la función $accion?\,(a, j)$ que retorna cierto si el jugador j tiene en su mano la carta de acción a y falso en caso contrario.

   Ejemplos:
   ```
   (accion? 'reversa (jugador 'quique UNO)) => #t
   (accion? 'comodin-color (cadr UNO)) => #f
   ```

   7.3. Definir mediante FOS la función $accion\text{-}x?\,(j)$ que retorna cierto si el jugador j tiene en su mano alguna carta de acción y falso en caso contrario.

   Ejemplos:
   ```
   (accion-x? (jugador 'pedro UNO)) => #f
   (accion-x? (car UNO)) => #t
   ```

   7.4. Definir mediante FOS la función $cartas\text{-}valor(valor, j)$ que retorna la lista de cartas de un $valor$ dado que tiene en su mano el jugador j.

   Ejemplos:
   ```
   (cartas-valor 'reversa (jugador 'quique UNO)) => ((reversa rojo))
   (cartas-valor 3 (cadr UNO)) => ((3 rojo) (3 rojo))
   ```

   7.5. Definir mediante FOS la función $cartas\text{-}color(color, j)$ que retorna la lista de cartas de un $color$ dado que tiene en su mano el jugador j. Deberá tenerse en cuenta que los comodines son cartas de cualquier color (o si se prefiere, de todos los colores posibles).

   Ejemplos:
   ```
   (cartas-color 'rojo (jugador 'maria UNO)) => ((0 rojo) (comodin-roba4))
   (cartas-color 'azul (cadr UNO)) => ((8 azul) (9 azul))
   ```

   7.6. Definir mediante FOS la función $compatibles(carta, j)$ que retorna la lista de cartas que el jugador j tiene en su mano y que son compatibles con la $carta$ dada. Una carta se considera compatible con otra cuando tienen el mismo valor o color (un comodín será compatible con cualquier carta). Proporciona la función solicitada para los casos siguientes:

a) Supuesto que $carta$ no es un comodín

b) Para cualquier valor de $carta$, incluidos los comodines

Ejemplo:
```
(compatibles '(0 azul) (jugador 'maria UNO))
    => ((0 rojo) (comodin-roba4) (3 azul))
```

7.7. Definir mediante FOS la función $jugadores\text{-}con\text{-}comodin(manos)$ que retorna la lista de nombres de jugadores que tienen en su mano un comodín. ***Ayuda***: define previamente la función $jugador\text{-}con\text{-}comodin(j)$ que retorna el nombre del jugador si en su mano tiene un comodín y la lista vacía en caso contrario.

Ejemplos:
```
(jugador-con-comodin (car UNO)) => ana
(jugadores-con-comodin UNO) => (ana luis maria)
```

7.8. Definir mediante FOS la función $jugadores\text{-}cartas\text{-}valor(valor, manos)$ que retorna las cartas del $valor$ dado que cada jugador tiene en la mano. No han de incluirse en esta lista los jugadores que no tengan carta alguna del $valor$ solicitado.

Ejemplo:
```
(jugadores-cartas-valor 3 UNO)
    => ((ana ((3 azul)))
        (rosa ((3 rojo) (3 rojo)))
        (maria ((3 verde) (3 azul)))
        (carmen ((3 amarillo)))
        (blanca ((3 amarillo))))
```

7.9. Definir mediante FOS la función $jugadores\text{-}cartas\text{-}color(color, manos)$ que retorna las cartas del $color$ dado que cada jugador tiene en la mano. No han de incluirse en esta lista los jugadores que no tengan carta alguna del $color$ solicitado.

Ejemplo:
```
(jugadores-cartas-color 'azul UNO)
    => ((ana ((comodin-color) (3 azul)))
        (rosa ((8 azul) (9 azul)))
        (luis ((comodin-color)))
        (maria ((comodin-roba4) (3 azul)))
        (carmen ((8 azul) (6 azul)))
        (blanca ((2 azul) (9 azul) (7 azul)))
        (quique ((1 azul))))
```