Programación Declarativa

Ingeniería Informática
Cuarto curso. Primer cuatrimestre.

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2019 – 2020

Práctica número 4.- Tipos compuestos de datos y funciones con argumentos obligatorios y opcionales

VECTORES

1. Escribe una función iterativa que calcule el **producto escalar** de dos vectores:
   - **Ejemplo**
     *(**productoEscalar** #(1 0 2) #(1 2 3))* ➔ *7*
     *(**productoEscalar** #(1 0 2 0 1) #(1 2 3 4 5))* ➔ *12*
   - **Observación**
     - Los dos vectores deben tener la misma longitud.

2. Codifica una función denominada ***aplicarMatriz*** que reciba un vector y una matriz y que cree otro vector cuyas componentes se obtengan haciendo el producto escalar del vector por cada una de las columnas de la matriz:
   - **Ejemplo**
     *(**aplicarMatriz** #(1 1 1) #(#(1 2) #(3 4) #(5 6))* ➔ *#(9 12)*
   - **Observación**
     - La longitud del vector que se pasa como parámetro debe ser igual al número de filas de la matriz.
     - La longitud del vector que crea la función debe ser igual al número de columnas de la matriz.

3. Escribe una función iterativa que calcule el **producto vectorial** de dos vectores del espacio $\mathbb{R}^3$
   - **Ejemplo**
     *(**productoVectorial** #(2 0 1) #(1 -1 3))* ➔ *#(1 -5 -2)*
   - **Observación**
     - Una vez codificada la función, comprueba que el vector obtenido es perpendicular a los vectores pasados como parámetros. Utiliza la función del producto escalar.

4. Codifica las siguientes funciones:
   a) Una función que calcule la media aritmética de los valores de un vector
      - **Ejemplo**

- *(**meanVector** #(2. 4. 8. 5. 7.))* ➔ *5.2*

b) Codifica una función denominada ***maxMin*** que reciba una matriz (no necesariamente cuadrada) y devuelva el máximo de los valores mínimos de las filas de la matriz.
- Ejemplo:
  - *(**maxMin** #(#(1. 2. 3.) #(4. 5. 6.) #(7. 8. 9.)))* ➔ *7.*

LISTAS

5. Codifica una función recursiva de cola que reciba una lista de números naturales y devuelva otra lista compuesta sólo por los números **primos**.
   - **Ejemplo**
     *(**filtrarPrimos** '(2 4 5 15 17 33))* ➔ *(2 5 17)*
   - **Observación:**
     - Utilícese un predicado auxiliar, denominado ***primo?*** que determine si un número natural es o no primo, para lo cual tendrá en cuenta que un número es primo si no tiene divisores menores o iguales que su raíz cuadrada.

6. Codifica una función **recursiva**, denominada ***suprimir***, que reciba como parámetro una lista de objetos, que puede tener sub-listas, y un elemento "x" y que cree como resultado otra lista en la que no aparezca dicho elemento "x".
   - Ejemplo
     *(**suprimir** '(a b d c (a b a) (d (e g) f) b) 'a)*
     ➔ (b d c (b) (d (e g) f) b)

7. Método de ordenación ***mergeSort***
   - **Descripción**
     - Datos de entrada : 5 4 1 3 2
       - ✓ División
         - Primera: 5 1 2 | 4 3
         - Segunda: 5 2 | 1 || 4 | 3
         - Tercera: 5 | 2 || 1 ||| 4 | 3
       - ✓ Fusión:
         - Primera: 2 5 | 1 || 3 4
         - Segunda: 1 2 5 | 3 4
         - Tercera: 1 2 3 4 5
   - Codifica una función que permita ordenar una **lista** de números utilizando el método ***mergeSort***.
     - Ejemplo
       - ✓ (***mergeSort*** '(5 4 1 3 2)) → (1 2 3 4 5)
   - **Observación**
     - Utiliza funciones auxiliares para la "división" (***split***):
       - ✓ Reparte los elementos de una lista en dos listas, dependiendo de que ocupen un "lugar" par o impar

- ✓ Ejemplos
  - (***split*** '()) → (() ())
  - (***split*** '(2)) → ((2) ())
  - (***split*** '(3 2)) → ((3) (2))
  - (***split*** '(1 3 2)) → ((1 2) (3))
  - (***split*** '(4 1 3 2)) → ((4 3) (1 2))
  - (***split*** '(5 4 1 3 2)) → → ((5 1 2) (4 3))
- y para la "fusión" (***merge***):
  - ✓ Une de forma ordenada dos listas ordenadas
  - ✓ Ejemplos
    - (***merge*** '() '()) → ()
    - (***merge*** '(1) '()) → (1)
    - (***merge*** '(1) '(2)) → (1 2)
    - (***merge*** '(1 3) '(2)) → (1 2 3)
    - (***merge*** '(1 3 5) '(2 4)) → (1 2 3 4 5)
    - (***merge*** '(1 2 3 5) '(4)) → (1 2 3 4 5)
    - (***merge*** '(1 2 3 4 5) '())→ (1 2 3 4 5)

FUNCIONES CON ARGUMENTOS OBLIGATORIOS Y OPCIONALES

8. Codifica una función denominada ***mergeSortDatos*** que permita ordenar una **cantidad variable** de números utilizando el método ***mergeSort***.
   - **Ejemplo**
     - (***mergeSortDatos***) → ()
     - (***mergeSortDatos*** 2) → (2)
     - (***mergeSortDatos*** 3 2) → (2 3)
     - (***mergeSortDatos*** 1 3 2) → 1 2 3)
     - (***mergeSortDatos*** 4 1 3 2) → (1 2 3 4)
     - (***mergeSortDatos*** 5 4 1 3 2) → (1 2 3 4 5)

9. **Comprobación de números ordenados**
   a) Codifica un predicado denominado ***listaOrdenada?*** que reciba una **lista** de números y compruebe si está ordenada ascendentemente
      - Ejemplos
        - *(**listaOrdenada?** '(1 2 3 4 5))* ➔ *#t*
        - *(**listaOrdenada?** '(3 1 2))* ➔ *#f*

   b) Codifica un predicado denominado ***ordenados?*** que reciba una cantidad variable de números y compruebe si están ordenados ascendentemente
      - Ejemplos
        - *(**ordenados?** 1 2 3 4 5)* ➔ *#t*
        - *(**ordenados?** 3 1 2)* ➔ *#f*

10. **Aplicar una función**
    a) Codifica una función denominada ***aplicarLista*** que
       - reciba dos parámetros

    - una función (procedimiento u operador)
    - y una **lista** de números
- y devuelva otra lista con el resultado de aplicar la función a cada elemento de la lista
- **Ejemplo**
    - (***aplicarLista*** sqrt '(1 2 3 4))
        → (1 1.4142135623730951 1.7320508075688772 2)
- **Observación**
    - Si la función es incorrecta o la lista está vacía, se devolverá una lita vacía
    - (***aplicarLista*** "a" '(1 2 3 4)) → ()
    - (***aplicarLista*** sqrt '()) → ()

b) Codifica una función denominada ***aplicar*** que
- reciba
    - un parámetro fijo: una función (procedimiento u operador)
    - y una cantidad variable de números
- y devuelva una lista con el resultado de aplicar la función a cada uno de los números recibidos como parámetros
- **Ejemplo**
    - *(**aplicar** sqrt 1 2 3 4)*
        → (1 1.4142135623730951 1.7320508075688772 2)
- **Observación**
    - Si la función es incorrecta o la lista está vacía, se devolverá una lita vacía
    - *(**aplicar** "a" 1 2 3 4) → ()*
    - *(**aplicar** sqrt) → ()*