Ciencias sociales computacionales

Práctica 2 – `R` básico

Sugerencias:

- Respete los nombres y cantidad de parámetros de cada función.
- Reutilice código de un ejercicio a otro; está permitido utilizar una función anterior para resolver los ítems subsiguientes en caso de que corresponda.
- Comente adecuadamente el código de cada función.
- Sea declarativo en la elección de los nombres de las variables.

Parte 1 – Problemas algebraicos

Ejercicio 1

Especifique e implemente las siguientes funciones:

a) `doble(n)`: devuelve el doble de `n`.

b) `signo(n)`: devuelve -1 si `n` es negativo, 0 si vale 0 y 1 si es positivo.

c) `abs(n)`: devuelve el valor absoluto de `n`.

d) `inversoMultiplicativo(n)`: devuelve el inverso multiplicativo de `n`.

e) `suma3(n1, n2, n3)`: devuelve la suma de `n1`, `n2` y `n3`.

f) `promedio3(n1, n2, n3)`: devuelve el promedio entre `n1`, `n2` y `n3`.

g) `maximo3(n1, n2, n3)`: devuelve el valor máximo entre `n1`, `n2` y `n3`.

h) `maximoAbsoluto3(n1, n2, n3)`: devuelve el máximo entre los valores absolutos de `n1`, `n2` y `n3`.

Ejercicio 2

Especifique e implemente las siguientes funciones *booleanas*:

a) `noEsCero(n)`: devuelve `True` si `n` es distinto de cero.

b) `iguales(n1, n2)`: devuelve `True` si `n1` es igual a `n2`.

c) `menor(n1, n2)`: devuelve `True` si `n1` es menor (estricto) a `n2`.

d) `par(n)`: devuelve `True` si `n` es un número par.

e) `divisible(n, d)`: devuelve `True` si `n` es divisible por `d`.

f) `imparDivisiblePorTresOCinco(n)`: devuelve `True` si `n` es divisible por 3 o por 5 pero no por 2.

Ejercicio 3

Especifique e implemente las siguientes funciones sobre enteros:

a) `factorial(n)`: devuelve el valor del factorial de `n`.

b) `sumaDivisores(n)`: devuelve la suma de todos los divisores positivos de `n`.

c) `primo(n)`: devuelve `True` si `n` es un número primo.

d) `menorDivisiblePorTres(n)`: dado un `n` positivo, devuelve el menor número mayor a `n` tal que sea divisible por 3.

e) `mayorPrimo(n1, n2)`: devuelve `True` si `n1` es el mayor primo que divide a `n2`.

f) `potencia(n1, n2)`: devuelve `True` si `n1` es una potencia de `n2`.

g) `mcd(n1, n2)`: devuelve el máximo común divisor entre `n1` y `n2`.

Parte 2 – Secuencias

Ejercicio 4

Especifique e implemente las siguientes funciones sobre secuencias. Para la implementación, en los casos en los que exista una función equivalente en `R`, no está permitido utilizarla.

a) `suma(a)`: devuelve la suma de todos los elementos de la lista `a`.

b) `promedio(a)`: devuelve el promedio de todos los elementos de la lista `a`. ¿Qué ocurre si `a` no tiene elementos?

c) `maximo(a)`: devuelve el máximo entre todos los elementos de la lista `a`.

d) `listaDeAbs(a)`: devuelve una lista con los valores absolutos de cada elemento de la lista `a`.

e) `maximoAbsoluto(a)`: devuelve el máximo entre los valores absolutos de todos los elementos de la lista `a`.

f) `divisores(n)`: devuelve una lista con todos los divisores positivos de `n`.

g) `cantidadApariciones(a, x)`: devuelve la cantidad de veces que aparece el elemento `x` en la lista `a`.

h) `masRepetido(a)`: devuelve el elemento que más veces aparece repetido en la lista `a`.

i) `todosPares(a)`: devuelve `True` si todos los elementos de la lista `a` son pares.

j) `ordenAscendente(a)`: devuelve `True` si todos los elementos de la lista `a` aparecen en orden ascendente. Ejemplos:

- `ordenAscendente([]) == True`
- `ordenAscendente(c(1,2,4)] == True`
- `ordenAscendente(c(4,2,1)) == False`

k) `reverso(a)`: devuelve una lista que cumple que sus elementos son los mismos que los de `a`, pero se encuentran en el orden inverso. Ejemplos:

- `reverso(c('h','o','l','a')) == c('a','l','o','h')`
- `reverso(reverso(c('h','o','l','a'))) == c('h','o','l','a')`

Ejercicio 5

Implemente funciones en `R` que cumplan con las siguientes especificaciones. Proponga un nombre declarativo en castellano para cada función implementada.

a) problema $\mathsf{A}(n : \mathbb{Z}) = x : \mathbb{R}\{$
   requiere : $n \geq 0$;
   asegura : $x^2 = n$;
$\}$

b) problema $\mathsf{B}(c(a : \mathbb{Z})) = x : \mathbb{Z}\{$
   asegura : $x = \left(\sum_{i=0}^{|a|-1} \beta\,(i \bmod 2 = 0) \cdot a[i] \right)$;
$\}$

c) problema $\mathsf{C}(c(a : \mathbb{Z})) = b : \mathbb{B}\{$
   asegura : $b = (\forall i : \mathbb{Z})(0 \leq i < |a| \rightarrow (a[i] = a[|a| - 1 - i]))$;
$\}$

d) problema $\mathsf{D}(c(a : \mathbb{Z})) = r : \mathbb{Z}\{$
   requiere : $|a| > 0$;
   asegura : $r = \left(\sum_{i=0}^{|a|-1} \beta\,(i \bmod 2 = 1) \cdot a[i] \right) / \left(\frac{|a|}{2} \right)$;
$\}$

e) problema $\mathsf{E}(c(a : \mathbb{Z})) = r : \mathbb{Z}\{$
   asegura : $(\exists i : \mathbb{Z})(0 \leq i \wedge i < |a| \wedge (\forall j : \mathbb{Z})(0 \leq j \wedge j < |a| \rightarrow a[i] \leq a[j]) \wedge r = a[i])$;
$\}$

f) problema $\mathsf{F}(c(a : \mathbb{Z})) = r : \mathbb{Z}\{$
   asegura : $(\exists i, j : \mathbb{Z})((0 \leq i \wedge i \leq j \wedge j < |a| \wedge (\forall k_1 : \mathbb{Z})(i \leq k_1 \wedge k_1 < j \rightarrow a[k_1] = a[i]) \wedge ((\forall l, m : \mathbb{Z})(0 \leq l \wedge l \leq m \wedge m < |a| \wedge (\forall k_2 : \mathbb{Z})(l \leq k_2 \wedge k_2 < m \rightarrow a[k_2] = a[i])) \rightarrow j - i \geq m - l) \wedge r = j - i)$;
$\}$

Ayuda: intente reescribir la especificación anterior utilizando predicados auxiliares con nombres declarativos.