

```

-- Grupo Nro 13 - Tarea 0
-- Integrantes
--     - Cesar Rodriguez
--     - Cesar Rodas
-- 20/08/2007
-- bugs o errores reportar a saddor@gmail.com
-- The authors disclaims the copyright over this code. You are legally
-- free to modify this code with a fee.
module Tarea0 where

--Ejercicio 0
--Divide un numero con resta recursiva
division a b
  | a < b      = 0
  | otherwise = 1 + ( division (a-b) b )

-- Ejercicio 1
-- Hallar el MCD por metodo normal y por metodo
-- "tradicional"
mcd_euclides a b
  | b == 0 = a
  | otherwise = mcd_euclides b (a `mod` b)

mcdinternal a b c
  | a < c || b < c                                = 1
  | a `mod` c == 0 && b `mod` c == 0              = c * mcdinternal (a `div` c) (b `div` c) c
  | otherwise                                      = mcdinternal a b (c+1)

mcd a b = mcdinternal a b 2

-- Ejercicio 2
-- primo' = calcula el numero primo'
--     a el numero que se desea saber si es primo o no
--     b es un contador (para la recursividad)
-- primo = es una interface para primo'
-- Funcionamiento:
--     ir dividiendo un numero N entre B (2..N). Si por el camino
--     se encontro una division con mod 0 no es primo. Si llego al
--     final (A == B) es primo.
--     El algoritmo se puede optimizar mas, pero funciona asi mismo.
primo' a b
  | a == b                = True
  | a `mod` b == 0 || a == 1 = False
  | otherwise             = primo' a (b+1)
primo n = primo' n 2

-- Ejercicio 3
-- hexaDigit n
--     retorna n (0..16) su digito hexa
-- insertaDerecha xs elem
--     Haskell tiene una peculiaridad que inserta elementos a una lista hacia la izq.
--     lo que por ejemplo 5:[1,2,3,4] = [5,1,2,3,4], la funcion inserta un elemento
--     hacia la derecha.
-- hexa
--     retorna un numero en base 10 a base 16
hexaDigit n = "0123456789ABCDEF"!!n -- que buenooo como lenguaje C
hexa n

```

```

| n < 16      = [ hexaDigit n ]
| otherwise   = insertDerecha (hexa (div n 16)) (hexaDigit (mod n 16))

```

```
insertDerecha xs elem
```

```

| length xs == 0 = [elem]
| otherwise      = head xs : insertDerecha (tail xs) elem

```

```
-- Ejercicio 4
```

```
-- Retorna un numero decimal a binario. Esta funcion retorna un entero, no una
-- cadena.
```

```
auxbinario n digito
```

```

| n == 1      = potencia 10 digito
| n == 0      = 0
| mod n 2 == 1 = potencia 10 digito + auxbinario (div n 2) (digito+1)
| otherwise   = auxbinario (div n 2) (digito+1)

```

```
binario n = auxbinario n 0
```

```
-- Ejercicio 5
```

```
-- Cuenta cuanto digitos 1 tiene una numero decimal al pasar a binario
```

```
binarioCont n
```

```

| n < 2      = n
| mod n 2 == 1 = 1 + binarioCont (div n 2)
| otherwise  = binarioCont (div n 2)

```

```
-- Ejercicio 6
```

```
-- Suma los elementos de una lista.
```

```
sumaLista xs
```

```

| length xs == 0      = 0
| otherwise           = h + sumaLista t

```

```
where
```

```

h = head xs
t = tail xs

```

```
-- Ejercicio 7
```

```
-- contaDig
```

```
-- cuenta los digitos que tiene un numero
```

```
-- _transBase
```

```
-- transforma un numero N de base BASE que tiene ORDEN digitos a decimal
```

```
-- transBase
```

```
-- transforma un numero N de base BASE a decimal, llamando a la funcion
```

```
-- _transBase
```

```
contDig n
```

```

| n <= 0      = 0
| otherwise   = 1 + contDig (div n 10)

```

```
_transBase n base orden
```

```

| orden < 0      = 0
| otherwise      = (digito * potencia base orden) + _transBase nnumero
base (orden-1)

```

```
where
```

```

digito      = div n xpot
nnumero    = n - digito * xpot
xpot       = potencia 10 orden

```

```
transBase n base = _transBase n base (contDig n)
```

```

-- Ejercicio 8
-- Cuenta la cantidad de minusculas que hay en una cadena
-- Gracias pablin por la ayuda en fromEnum, aunque se tendria que poder hacer tb
-- sin fromEnum.
-- fromEnum transforma una letra en su valor ASCII
contMin xs
  | length xs == 0 = 0
  | fromEnum h >= fromEnum 'a' && fromEnum h <= fromEnum 'z' = 1 + contMin t
  | otherwise = contMin t
  where
    h = head xs
    t = tail xs

-- Ejercicio 9
-- Invierte el orden de una lista
-- [1,2,3,3] = [3,3,2,1]
invertir xs
  | length xs == 0 = []
  | otherwise = insertDerecha (invertir (tail xs) ) (head xs)

-- Ejercicio 10
-- Entrada "A01B02C03D04E06" 'B'
-- Busca B si existe retorna 02
extraer xs pos cnt e i
  | cnt >= i || length xs == 0 = []
  | pos >= e = h: extraer t (pos+1) (cnt+1) e i
  | otherwise = extraer t (pos+1) cnt e i
  where
    h = head xs
    t = tail xs

ejer10 xs f
  | length xs == 0 = "no existe" -- no existe
  | f == elemMitad = extraer xs 0 0 (mitad+1) 2 --encontro!
  | f > elemMitad = ejer10 (extraer xs 0 0 (mitad+3) mitad) f --parte derecha
  | f < elemMitad = ejer10 (extraer xs 0 0 0 (mitad)) f --parte izq.
  where
    elemMitad = xs!!mitad
    mitad = ( div (length xs) 2 ) - 1

-- Ejercicio 11
-- Potencia por multiplicacion recursiva
potencia a b
  | b == 0 = 1
  | b > 0 = a * potencia a (b-1)

```