module Lab01 where

2

3

import Data.List

4

5

{-

6

1) Corregir los siguientes programas de modo que sean aceptados por GHCi.

7

-}

8

9

-- a)

10

not b = case b of

11

True -> False

12

False -> True

13

14

-- b)

15

in [x] = []

16

in (x:xs) = x : in xs

17

in [] = error "empty list"

18

19

-- c)

20

Length [] = 0

21

Length (\_:l) = 1 + Length l

22

23

-- d)

24

list123 = (1 : 2) : 3 : []

25

26

-- e)

27

[] ++! ys = ys

28

(x:xs) ++! ys = x : xs ++! ys

29

30

-- f)

31

addToTail x xs = map +x tail xs

32

33

-- g)

34

listmin xs = head . sort xs

35

36

-- h) (\*)

37

smap f [] = []

38

smap f [x] = [f x]

39

smap f (x:xs) = f x : smap (smap f) xs

40

41

{-

42

2. Definir las siguientes funciones y determinar su tipo:

43

44

a) five, que dado cualquier valor, devuelve 5

45

46

b) apply, que toma una función y un valor, y devuelve el resultado de

47

aplicar la función al valor dado

48

49

c) ident, la función identidad

50

51

d) first, que toma un par ordenado, y devuelve su primera componente

52

53

e) derive, que aproxima la derivada de una función dada en un punto dado

54

55

f) sign, la función signo

56

57

g) vabs, la función valor absoluto (usando sign y sin usarla)

58

59

h) pot, que toma un entero y un número, y devuelve el resultado de

60

elevar el segundo a la potencia dada por el primero

61

62

i) xor, el operador de disyunción exclusiva

63

64

j) max3, que toma tres números enteros y devuelve el máximo entre llos

65

66

k) swap, que toma un par y devuelve el par con sus componentes invertidas

67

-}

68

69

{-

70

3) Definir una función que determine si un año es bisiesto o no, de

71

acuerdo a la siguiente definición:

72

73

año bisiesto 1. m. El que tiene un día más que el año común, añadido al mes de febrero. Se repite

74

cada cuatro años, a excepción del último de cada siglo cuyo número de centenas no sea múltiplo

75

de cuatro. (Diccionario de la Real Academia Espaola, 22ª ed.)

76

77

¿Cuál es el tipo de la función definida?

78

-}

79

80

{-

81

4)

82

83

Defina un operador infijo \*$ que implemente la multiplicación de un

84

vector por un escalar. Representaremos a los vectores mediante listas

85

de Haskell. Así, dada una lista ns y un número n, el valor ns \*$ n

86

debe ser igual a la lista ns con todos sus elementos multiplicados por

87

n. Por ejemplo,

88

89

[ 2, 3 ] \*$ 5 == [ 10 , 15 ].

90

91

El operador \*$ debe definirse de manera que la siguiente

92

expresión sea válida:

93

94

-}

95

96

v = [1, 2, 3] \*$ 2 \*$ 4

97

98

99

{-

100

5) Definir las siguientes funciones usando listas por comprensión:

101

102

a) 'divisors', que dado un entero positivo 'x' devuelve la

103

lista de los divisores de 'x' (o la lista vacía si el entero no es positivo)

104

105

b) 'matches', que dados un entero 'x' y una lista de enteros descarta

106

de la lista los elementos distintos a 'x'

107

108

c) 'cuadrupla', que dado un entero 'n', devuelve todas las cuadruplas

109

'(a,b,c,d)' que satisfacen a^2 + b^2 = c^2 + d^2,

110

donde 0 <= a, b, c, d <= 'n'

111

112

(d) 'unique', que dada una lista 'xs' de enteros, devuelve la lista

113

'xs' sin elementos repetidos

114

-}

115

116

unique :: [Int] -> [Int]

117

unique xs = [x | (x,i) <- zip xs [0..], not (elem x (take i xs))]

118

119

{-

120

6) El producto escalar de dos listas de enteros de igual longitud

121

es la suma de los productos de los elementos sucesivos (misma

122

posición) de ambas listas. Definir una función 'scalarProduct' que

123

devuelva el producto escalar de dos listas.

124

125

Sugerencia: Usar las funciones 'zip' y 'sum'. -}

126

127

{-

128

7) Definir mediante recursión explícita

129

las siguientes funciones y escribir su tipo más general:

130

131

a) 'suma', que suma todos los elementos de una lista de números

132

133

b) 'alguno', que devuelve True si algún elemento de una

134

lista de valores booleanos es True, y False en caso

135

contrario

136

137

c) 'todos', que devuelve True si todos los elementos de

138

una lista de valores booleanos son True, y False en caso

139

contrario

140

141

d) 'codes', que dada una lista de caracteres, devuelve la

142

lista de sus ordinales

143

144

e) 'restos', que calcula la lista de los restos de la

145

división de los elementos de una lista de números dada por otro

146

número dado

147

148

f) 'cuadrados', que dada una lista de números, devuelva la

149

lista de sus cuadrados

150

151

g) 'longitudes', que dada una lista de listas, devuelve la

152

lista de sus longitudes

153

154

h) 'orden', que dada una lista de pares de números, devuelve

155

la lista de aquellos pares en los que la primera componente es

156

menor que el triple de la segunda

157

158

i) 'pares', que dada una lista de enteros, devuelve la lista

159

de los elementos pares

160

161

j) 'letras', que dada una lista de caracteres, devuelve la

162

lista de aquellos que son letras (minúsculas o mayúsculas)

163

164

k) 'masDe', que dada una lista de listas 'xss' y un

165

número 'n', devuelve la lista de aquellas listas de 'xss'

166

con longitud mayor que 'n' -}