UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

Paradigmas de Programación

TRABAJO PRÁCTICO N° 8

Programación Funcional

1. Haskell: Ejercicios Básicos

Ejercicio 1.1. Definir las siguientes funciones y evaluar cada una de ellas. Se pueden reutilizar las funciones que ya han implementado en pasos anteriores para armar nuevas funciones.

A. El volumen de una esfera

B. La suma de monedas tal que ($sumaCoins$ a b c d e) es la suma de los pesos correspondientes a: a monedas de 1 centavo, b de 5 centavos, c de 10 centavos, d de 50 centavos, e de 1 peso.

C. Incrementar todos los elementos de una tupla de tres enteros.

D. Calcular el cuadrado de un número.

E. calcular el valor de un número elevado a la 4 utilizando la función del inciso anterior

F. Calcular la media aritmética de tres valores numéricos.

G. Calcular el máximo entre 3 números, puede utilizar la función predefinida max

H. Calcular el máximo entre 6 números, puede utilizar la función predefinida max

I. Definir la función $area$ tal que la función área a b c es el área de un triangulo de lados a,b,c.

J. Definir una función tal que reciba un par (x, y) y retorne el cuadrante, puede informar mediante mensaje o devolviendo el número de cuadrante. El par no esta sobre los ejes x ni y.

K. La función $igualesTres$ verifica que tres elementos x, y y z son iguales.

L. La función $diferentesTres$ verifica que tres elementos x, y y z son diferentes.

M. La función $igualesCuatro$ verifica que tres elementos v, x, y y z son iguales utilizando la función definida en el punto D.

N. Definir las raíces de una ecuación de segundo grado

Ñ. Determinar si un año es bisiesto

O. Definir el operador XOR

2. Listas

Ejercicio 2.1. Desarrolle los siguientes ejercicios sobre listas:

A. Definir la función divisores tal que $divisores\ x$ es la lista de los divisores de x

```
divisores 4 == [1,-1,2,-2,4,-4]
divisores (-6) == [1,-1,2,-2,3,-3,6,-6]
```

B. Dada una cantidad de segundos, devuelve la cantidad de horas, minutos y segundos equivalente.

C. Definir la función primo tal que primo x se verifica si x es primo

D. Definir una función que devuelva una listas de primos hasta un valor N

E. Definir la función $tomar$ tal que tomar n l es la lista de los n primeros elementos de l.

F. Redefinir la función $tomarMientras$ tal que tomarMientras p l es la lista de los elementos iniciales de l que verifican el predicado p

G. Definir la función $nIndex$ tal que $nIndex$ l n es elemento enésimo de l, empezando a numerar con el 0.

H. Redefinir la función $elem$ tal que $elem\ e\ l$ se verifica si e es un elemento de l.

I. Definir la función que convierte el número decimal en su correspondiente número binario. Los números binarios deben almacenarse como una lista

J. Definir un nuevo tipo de datos para un número complejo y algunas operaciones básicas como la suma y la multiplicación de números complejos

K. Definir al menos 10 colores definiendo su propio tipo de datos. Ingresar la lista y evaluar si la lista tiene los colores predefinidos por el usuario.

L. Teniendo en cuenta lo definido en el punto anterior, armar una función que permita ingresar 2 colores y devolver el nuevo color que se genera. Puede ser un string el nuevo color.

M. Crear una función ocurrencias, que toma un elemento y una lista y devuelve el número de ocurrencias del elemento en la lista.

N. Escribir un programa que genere todas las permutaciones de n objetos diferentes (se ingresan los datos como una lista).

Ñ. Crear una lista que contenga todos los enteros dentro de un rango dado

O. Generar las combinaciones de K objetos distintos elegidos de los N elementos de una lista

P. mapToSucesor: dada una lista de enteros, devuelve la lista de los sucesores de cada entero.

Q. filtrarPositivos: dada una lista de enteros, devuelve una lista con los elementos que son positivos.

R. reversa': dada una lista de enteros, devuelve la lista con los mismos elementos de atrás para adelante.

3. Combinación de tipos

Ejercicio 3.1. Definir la firma y el cuerpo de los siguientes enunciados:

A. sumaPar: dada una lista de pares, devuelve una nueva lista en la que cada elemento es la suma de los elementos de cada par.

B. zipMaximos: dadas dos listas de enteros, devuelve una lista donde el elemento n es el máximo entre el elemento n de la lista 1 y de la lista 2.

C. zipSort: dadas dos listas de enteros de igual longitud, devuelve una lista de pares (min,max), donde min y max son el mínimo y el máximo entre los elementos de ambas listas en la misma posición.

D. takePersonas: dada una lista de Personas [nombre, apellido y fecha de nacimiento] (también declare un tipo de dato Date) ordenada ascende*+ntemente por fecha de nacimiento; y una fecha, devuelve el segmento más largo de la lista con las personas que nacieron antes dicha fecha.1

E. $dropPrecio$: dada una lista de Pizzas [lista de ingredientes y precio] en orden ascendente por precio, $dropPrecio$ devuelve el segmento más largo de la lista que comienza con la pizza que tiene el menor precio superior a $200

F. takeNombresPersonas: dada una lista de Personas y una fecha devuelve los nombres de las personas incluidas en segmento más largo de la lista con las personas que nacieron antes dicha fecha.

G. Definir una función que permita indicar que nota saco el alumno en base al número obtenido en el parcial.

   a) Utilice una función con guardas

   b) Utilice una función con patrones

   c) Utilice un tipo de datos definidos por el usuario con las notas que es posible sacarse. Utilizarlo en el ejercicio

H. En Geometría euclidiana plana recibe el nombre de cuadrante cada una de las cuatro regiones infinitas en que los ejes del Sistema Cartesiano bidimensional dividen al plano. Definir un tipo de dato que contenga los cuadrantes, el origen, los ejes cartesianos, es decir los posibles espacios donde un punto puede estar localizado en un eje cartesiano. Posteriormente defina la función cuadrante que permita dado dos puntos mostrar el cuadrante a donde pertenece.

I. Definir el tipo de dato Animal y una función que muestre un mensaje respecto al animal.

4. Funciones Recursivas

Ejercicio 4.1. Sin usar funciones de alto orden ni funciones definidas en *prelude*, definir recursivamente las siguientes funciones y determine su tipo más general:

A. $suma$, que suma todos los elementos de una lista de números

B. $alguno$, que devuelve True si algún elemento de una lista de valores booleanos es True, y False en caso contrario

C. $todos$, que devuelve True si todos los elementos de una lista de valores booleanos son True, y False en caso contrario

D. $codigos$, que dada una lista de caracteres, devuelve la lista de sus ordinales

E. $restos$t, que calcula la lista de los restos de la división de los elementos de una lista de números dada por otro número dado

F. Incrementar todos los elementos de una lista de enteros.

G. $cuadrados$, que dada una lista de números, devuelva la lista de sus cuadrados

H. $longitudes$, que dada una lista de listas, devuelve la lista de sus longitudes

I. $orden$, que dada una lista de pares de números, devuelve la lista de aquellos pares en los que la primera componente es menor que el triple de la segunda

J. $pares$, que dada una lista de enteros, devuelve la lista de los elementos pares

K. $letras$, que dada una lista de caracteres, devuelve la lista de aquellos que son letras (minúsculas o mayúsculas)

L. $masDe$, que dada una lista de listas xs y un número n, devuelve la lista de aquellas listas de xs con longitud mayor que n

M. Definición de un operador que aplica una lista de funciones a un entero y devuelve la lista de enteros de los resultados.

N. Definir la función $deEnteroACadena$ tal que $deEnteroACadena$ n es la cadena correspondiente al número entero n.

Ñ. Definir una función que devuelva la posición inicial de una sublista en una lista dada. Por ejemplo:

```
ghci> findList [1] [4,5,1,2,5,1]
2 --resultado, recordar que empieza desde 0 una lista
```

ghci>findList [7] [4,5,1,2,5,1] Nothing –resultado, puede definir su propio tipo para devolver resultados

5. Tipos de funciones

Ejercicio 5.1. Inferir, de ser posible, los tipos de las siguientes funciones:

A.
```
incLista [] = []
incLista (x:xs) = (x + 1) : incLista xs
```

B. `fc = \a -> (\b -> 2 * a + b)`

C. `impar = not . even`

D. `removeall = filter . (/=)`

E. `&&`

F. `modulus = sqrt . sum . map square;`

6. Teoría

Ejercicio 6.1. Qué es la programación funcional

Ejercicio 6.2. Qué caracteristicas presenta un paradigma funcional

Ejercicio 6.3. Que es un programa escrito en un lenguaje funcional? Que rol cumple la computadora?

Ejercicio 6.4. Cual es el concepto de variables en los lenguajes funcionales?

Ejercicio 6.5. Diferencias entre un lenguaje funcional y un lenguaje orientado a objetos

Ejercicio 6.6. En que situaciones sería mejor utilizar el paradigma funcional ?

Ejercicio 6.7. El paradigma funcional se basa en el funcionamiento de la maquina Von Neumman?

Ejercicio 6.8. Las monadas nos permiten solventar diferentes problemas. ¿Cuales serían ellos? Justifique como estos problemas afectan a un lenguaje funcional.