**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

**Paradigmas de programación**

**Prof.: Roberto Tecla Parra**

**Alumno: JESUS ALBERTO DORIAN LUNA MARTINEZ**

**Grupo: 3CM3**

**Guía 2do Parcial**

**GUIA DE PARADIGMAS DE PROGRAMACION SEGUNDO PARCIAL**

**1.-En haskell una cadena es**

a) un arreglo de caracteres

b) una lista de caracteres

c) un apuntador a caracter

d) una lista de enteros

**2.-Cual es la funcion fmap para las funciones**

a) (.) b) map c) Just (f x) d) Right (f x)

**3.-Cual es la funcion fmap para las listas**

a) (.) b) map c) Just (f x) d) Right (f x)

**4.-Just es la funcion pure para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**5.-[x] es la funcion pure para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**6.-(\\_ -> x) es la funcion pure para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**7.-fmap f algo es la funcion (<\*>) para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**8.-[f x | f <- fs, x <- xs] es la funcion (<\*>) para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**9.-\x -> f x (g x) es la funcion (<\*>) para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**10a.-Cual es el tipo de foldl**

I.- (a -> b) -> [a] -> [ b]

II.- (a -> b) -> f a -> f b

III.- f (a -> b) -> f a -> f b

IV.- m a -> (a -> m b) -> m b

V.- (a -> Bool) -> [a] -> [a]

VI.-(a -> b -> a) -> a -> [b] -> a

**10b.-Cual es el tipo de map**

I.- (a -> b) -> [a] -> [ b]

II.- (a -> b) -> f a -> f b

III.- f (a -> b) -> f a -> f b

IV.- m a -> (a -> m b) -> m b

V.- (a -> Bool) -> [a] -> [a]

VI.-(a -> b -> a) -> a -> [b] -> a

**11.-Cual es el tipo de fmap**

I.- (a -> b) -> [a] -> [ b]

II.- (a -> b) -> f a -> f b

III.-f (a -> b) -> f a -> f b

IV.- m a -> (a -> m b) -> m b

V.- (a -> Bool) -> [a] -> [a]

VI.-(a -> b -> a) -> a -> [b] -> a

**12.-Cual es el tipo de la funcion (<\*>)**

I.- (a -> b) -> [a] -> [ b]

II.- (a -> b) -> f a -> f b

III.-f (a -> b) -> f a -> f b

IV.- m a -> (a -> m b) -> m b

V.- (a -> Bool) -> [a] -> [a]

VI.-(a -> b -> a) -> a -> [b] -> a

**13.-Cual es el tipo de la funcion (>>=)**

I.- (a -> b) -> [a] -> [ b]

II.- (a -> b) -> f a -> f b

III.-f (a -> b) -> f a -> f b

IV.- m a -> (a -> m b) -> m b

V.- (a -> Bool) -> [a] -> [a]

VI.-(a -> b -> a) -> a -> [b] -> a

**14.-Cual es el kind de Int**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**15.-Cual es el kind de Maybe Int**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**16.-Cual es el kind de Maybe**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**17.-Cual es el kind de Either Int**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**18.-Cual es el kind de Either Int String**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**19.-Cual es el kind de Either**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**20.-Cual es el kind de un Functor**

a) \* b) \* -> \* c) \* -> \* -> \* d) \* -> \* -> \* -> \*

**21.-En Haskell que se usa para los computos no deterministas**

a) las listas b) los Maybes c) las funciones d) los Eithers

**22.-Cual es el elemento neutro para Maybe**

a) 0 b) Nothing c) [] d) 1 e) id

**23.-Cual es el elemento neutro para las listas**

a) 0 b) Nothing c) [] d) 1 e) id

**24.-Cual es la funcion mappend para Maybe**

a) (.) b) map c) Just (m1 `mappend` m2) d) ++ e) (:)

**25.-Cual es la funcion mappend para las listas**

a) (.) b) map c) Just (f x) d) ++ e) (:)

**26.- Just es la funcion return para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**27.- [x] es la funcion return para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers

**28.- f x es la funcion (>>=) para \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

a) las listas b) los Maybes c) las funciones d) los Eithers\*\*

PROBLEMAS DE HASKELL:

**Problema 1.-Escriba la funcion de abajo en el estilo point-free**

**suma x = foldr (+) 0 x**

***R******:*** *import Control.Applicative*

*suma=foldr (+) 0*

**Problema 2 .-Codificar en Haskell una funcion que divida 2 numeros y cuyo tipo de**

**retorno sea Maybe Double**

***R******:*** *dividir::(Integral a)=>a->a->Maybe a*

*dividir x y*

*| y==0 = Nothing*

*| otherwise = Just (div x y)*

**Problema 3 .-Usar el estilo aplicativo para codificar una funcion que multiplique**

**dos Maybe Int**

**Hint : use pure, (+), y <\*>**

***R:****multiplicar::Maybe Int->Maybe Int->Maybe Int*

*multiplicar x y = pure (\*) <\*> x <\*> y*

**Problema 4 .-Usar el estilo aplicativo como remplazo para la lista por comprensión de abajo.**

**[ x\*y | x <- [2,5,10], y <- [8,10,11]]**

**Hint : use <\*>, (\*), y <$>**

***R:****[(\*2), (\*5), (\*10)] <\*> [8, 10, 11]*

**Problema 5 .-Que da como resultado la expresion de abajo?**

**[(\*2),(+20),(^3)] <\*> [6,10, 16]**

***R:*** *[12, 20, 32, 26, 30, 36, 216, 1000, 4096]*

**Problema 6 .- Dado el codigo de ZipList mostrado abajo**

**instance Applicative ZipList where**

**pure x = ZipList (repeat x)**

**ZipList fs <\*> ZipList xs = ZipList (zipWith (\f x -> f x) fs xs)**

**¿Que da como resultado la expresion de abajo?**

**ZipList [(+1),(\*2), (/3), (\x -> 3\*x+50)] <\*> ZipList [100,100..]**

***R:*** *[101, 200, 3.3333, 350]*

**Problema 7 .-Dada la función sequenceA de abajo**

**sequenceA :: (Applicative f) => [f a] -> f [a]**

**sequenceA [] = pure []**

**sequenceA (x:xs) = (:) <$> x <\*> sequenceA xs**

**Use la función sequenceA como remplazo para la lista por comprensión de abajo.**

**[[x,y] | x <- [10,12], y <- [35, 80]]**

***R:*** *sequenceA [[10, 12], [35, 80]]*