{-Algoritmo y Estructura de Datos I

Proyecto 2

Greiff, Alejandro

Vispo, Valentina-}

--1--

--a)

data Carrera = Matematica | Fisica | Computacion | Astronomia | Profesorado deriving (Eq,Show)

--b)

titulo::Carrera -> String

titulo Matematica= "Licenciatura en Matematica"

titulo Fisica= "Licenciatura en Fisica"

titulo Computacion= "Licenciatura en Computacion"

titulo Astronomia= "Licenciatura en Astronomia"

titulo Profesorado= "Profesorado de... No se"

--c)

{-Si se puede hacer con analisis por casos pero se debe decir que las

carreras son comparables, para eso el siguiente ejemplo

-}

titulo'::Carrera -> String

titulo' x| x==Matematica = "Licenciatura en Matematica"

|otherwise = "Qcy"

--2--

--Ingreso es un sinonimo de tipo

type Ingreso = Int

--a)

--tipo enumerado

data Funcion = Teorico | Practico deriving (Eq,Show)

--tipo algebraico

data Rol = Decanx --constructor sin argumento

| Docente Funcion --constructor con un argumento

| Estudiante Carrera Ingreso --contructor con dos argumento

deriving (Eq,Show)

--b)

--Docente es de tipo Funcion -> Rol

--c)

cuantos\_doc::[Rol]->Funcion->Int

cuantos\_doc [] c = 0

cuantos\_doc (x:xs) c |x == Docente c = 1 + cuantos\_doc xs c

|otherwise = cuantos\_doc xs c

--d)

cuantos\_doc'::[Rol]->Funcion->Int

cuantos\_doc' xs c = length (filter (==Docente c) xs)

--e)

data Rol'=Decane Genero

data Genero= Hombre|Mujer|Helicoptero\_Apache

--f)

data Rol''= Estudiante' Carrera Carrera Ingreso

--estudia::Rol''-> Carrera-> Bool

--estudia x c| x==Estudiante' c a i || x== Estudiante' a c2 i2= True

-- | otherwise =False

--3--

--a)

data Persona= Per String Int Int Rol deriving (Eq,Show)

--b)

{-Noh

-}

--c)

--(1)

edad::Persona->Int->Int

edad (Per \_ \_ e \_) a = a-e

--(2)

existe::String->[Persona]->Bool

existe a [] = False

existe a ((Per b \_ \_ \_):xs)| a == b = True

|otherwise = existe a xs

est\_astronomia::[Persona]->[Persona]

est\_astronomia []=[]

est\_astronomia ((Per a b c (Estudiante Astronomia d)):xs)=

(Per a b c (Estudiante Astronomia d)): est\_astronomia xs

est\_astronomia (p:ps) = est\_astronomia ps

--(3)

padron\_docente::[Persona]->[(String,Int)]

padron\_docente []=[]

padron\_docente ((Per a b c (Docente n)):ps)=(a,b): padron\_docente ps

padron\_docente (p:ps)= padron\_docente ps

--4--

data Cola= Vacia | Encolada Persona Cola deriving (Eq,Show)

--a)

--(1)

atender::Cola->Cola

atender (Encolada p c)= c

--(2)

encolar::Persona-> Cola -> Cola

encolar (Per n d f r) (Vacia) = (Encolada (Per n d f r) Vacia)

--encolar (Per n d f r) (Encolada \_ c) = encolar (Per n d f r) c

encolar (Per n d f r) (Encolada p c) = Encolada (Per n d f r) (encolar (Per n d f r) c)

encolar' :: Persona-> Cola -> Cola

encolar' ultPer Vacia = Encolada ultPer Vacia

encolar' ultPer (Encolada per c) = Encolada per (encolar ultPer c)

-- Agrega a una persona en el primer lugar de la cola

--encolar::Persona->Cola->Cola

--encolar (Per n d f r) (Encolada p c)=(Encolada (Per n d f r) (Encolada p c))

--(3)

busca::Cola->Funcion->Persona

busca (Encolada (Per n d f (Docente x)) c) i | (Per n d f (Docente x)) ==

(Per n d f (Docente i))=

(Per n d f (Docente x))

|otherwise = busca c i

busca (Encolada (Per n d f (k)) c) i= busca c i