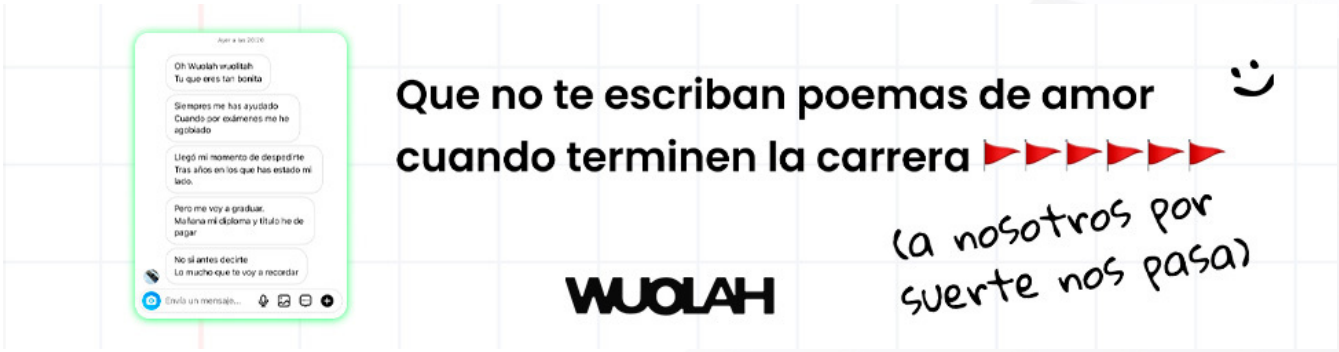
Examen parcial resuelto ED

jmp__0807

Estructuras de Datos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga



Que no te escriban poemas de amor cuando terminen la carrera ►►►►►►

(a nosotros por suerte nos pasa) ☺

WUOLAH

TAD POLINOMIO

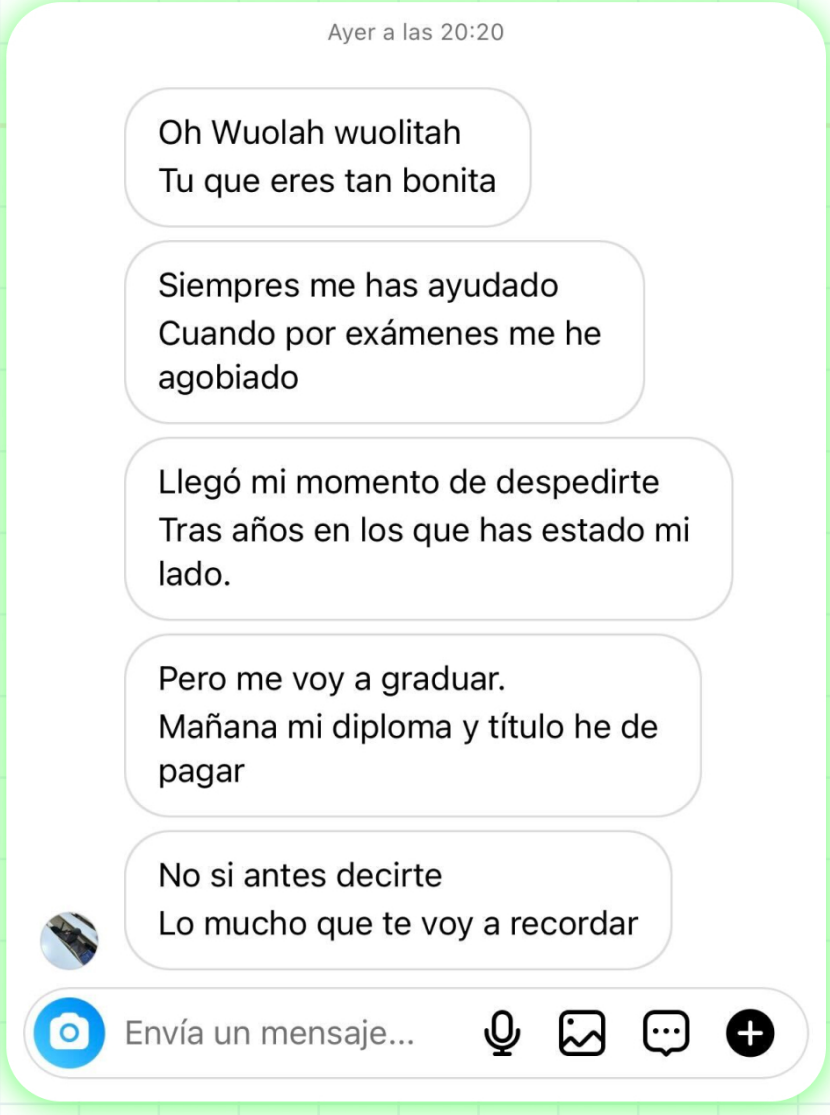
Estructura de Datos, grupo E. Curso 2021-2022

ESPECIFICACIÓN INFORMAL

- Un polinomio es expresión formada por una secuencia de términos que tienen un coeficiente (distinto de cero) diferente grado.
- Por ejemplo el polinomio $x^3 + 2x^2 + 8$ está compuesto de 3 términos:
  - Grado 0 coeficiente 8
  - Grado 2 coeficiente 2
  - Grado 3 coeficiente 1

Que no te escriban poemas de amor cuando terminen la carrera 🚩🚩🚩🚩🚩🚩

(a nosotros por suerte nos pasa)



WUOLAH

IMPLEMENTACIÓN

Vamos a implementar un módulo Haskell para trabajar con polinomio.

Utilizaremos el tipo de datos algebraico Pol

```
data Pol = Nil | P Grade Coefficient (Pol) deriving Show
```

El polinomio se representa como una secuencia de términos ordenados de mayor a menor grado.

Invariante: Todos los términos tienen coeficiente distinto de 0

Todos los términos tiene diferente grado

`Nil` representa un polinomio que no tiene ningún término (grado 0 coeficiente 0)

`P 3 1 (P 2 2 (P 0 8 Nil))` representa el polinomio $x^3 + 2x^2 + 8$

Que no te escriban poemas de amor
cuando terminen la carrera ►►►►►►

(a nosotros por
suerte nos pasa)

WUOLAH

IMPLEMENTACIÓN

Operaciones disponibles:

empty :: Pol

grade :: Pol -> Grade

coeff :: Grade -> Pol -> Coefficient

insert :: Grade -> Coefficient -> Pol -> Pol

remove :: Grade -> Pol -> Pol

list2Pol :: [Integer] -> Pol

sumP :: Pol -> Pol -> Pol

foldPol::(Grade -> Coefficient -> b -> b) -> b -> Pol -> b

eval :: Integer -> Pol -> Integer

```haskell
-- Javier Montes Perez TAD Polinomio

import Test.QuickCheck

type Grade = Int
type Coefficient = Integer

data Pol = Nil | P Grade Coefficient (Pol) deriving Show

s1 = P 2 1 (P 1 3 (P 0 8 Nil)) -- x^2 + 3x + 8
s2 = P 3 2 (P 1 2 (P 0 10 Nil))

-- Ejercicio 1 -> 0.05 ptos
-- Devuelve un polimonio vacio

empty :: Pol
empty = Nil

-- Ejercicio 2 -> 0.05 ptos
-- Devuelve el grado maximo del polinomio

grade :: Pol -> Grade
grade Nil = error "grade on nil polynomial"
grade (P g c rp) = g

-- Ejercicio 3 -> 0.05 ptos
-- Devuelve el coeficiente del termino de grado g

coeff :: Grade -> Pol -> Coefficient
coeff _ Nil = error "coeff on nil polynomial"
coeff g1 (P g2 c rp)
    | g1 > g2  = error "el coeficiente no se encuentra en este polinomio"
    | g1 == g2 = c
    | otherwise = coeff g1 rp

-- Ejercicio 4 -> 0.2 ptos
-- Añade el termino de grado g con coeficientec al polinomio
-- Si el polinomio ya tenia un termino de ese grado, actualiza el coeficiente
-- es decir, reemplaza el antiguo coeficiente por el nuevo

insert:: Grade -> Coefficient -> Pol -> Pol
insert _ 0 p = p
insert g c Nil = (P g c Nil)
insert g c r@(P g' c' rp)
    | g > g'    = (P g c r)
    | g == g'   = (P g c rp)
    | otherwise = (P g' c' (insert g c rp))

-- Ejercicio 5 -> 0.2 ptos
-- Elimina el termino g del polinomio
-- Si el polinomio no tenía terminos de grado g no hace nada

remove :: Grade -> Pol -> Pol
remove _ Nil = Nil
remove g (P g' c' rp)
    | g > g' = (P g' c' rp)
    | g == g' = rp
```

```haskell
    | otherwise = P g' c' (remove g rp)

-- Ejercicio 5 (0.2 ptos con plegado de listas / 0.15 sin plegado): Dada una lista con los
-- coeficientes de los terminos de un polinomio, donde la posición indica el grado,
-- devuelve el Pol correspondiente.

-- Si la lista tienen un coeficiente 0 ese termino no se almacena en la lista. Por ejemplo
-- (x^3 + 2x^2 + 8) se representa con la lista [8, 0, 2, 1] y usando list2Pol se obtiene
-- el polinomio P 3 1 (P 2 2 (P 0 8 Nil))

list2Pol :: [Integer] -> Pol
list2Pol [] = Nil
list2Pol l = foldr (\(i,x) solResto -> insert i x solResto) Nil (zip [0..] l)

-- Ejercicio 7 (0.15 ptos):suma dos polinomios que pueden tener diferente grado

sumP :: Pol -> Pol -> Pol
sumP p Nil = Nil
sumP Nil p = Nil
sumP p1@(P g1 c1 r1) p2@(P g2 c2 r2)
    | g1 > g2  = P g1 c1 (sumP r1 p2)
    | g2 > g1  = P g2 c2 (sumP p1 r2)
    | g1 == g2 = P g1 (c1+c2) (sumP r1 r2)

-- Ejercicio 8 (0.15 ptos) : plegado a la derecha de polinomios

foldPol :: (Grade -> Coefficient -> b -> b) -> b -> Pol -> b
foldPol = undefined -- ni puta idea

-- Ejercicio 9 (0.1 ptos con plegado de Pol 0.05 sin plegado): evaluacion de el polinomio.
-- Por ejemplo, si tenemos el polinomio x^2 +1 representado como
-- p1 = P 2 1 (P 0 1 Nil) eval 3 p1 devuelve el resultado de evaluar el polinomio en 3 (3^2 + 1 = 10)

eval :: Integer -> Pol -> Integer
eval _ Nil = 0
eval n (P g c rp) = (c*n^g) + eval n rp

-- Ejercicio 10 (0.1 ptos):
-- Establece una propiedad quickCheck que compruebe que la suma de dos polinomios p1 y p2
-- tiene el mismo grado que el polinomio de mayor grado

p1_suma :: Pol -> Pol -> Bool
p1_suma first@(P g1 c1 r1) second@(P g2 c2 r2) = g1 ≥ g2 ⟹ grade (sumP first second) == g1
```


si lees esto me debes un besito