# Estructuras Discretas
# Proyecto 2

Odín Miguel Escorza Soria  Daniela Calderón Pérez

Facultad de Ciencias UNAM
Fecha de entrega: 16 noviembre 2017

## Dados los siguientes tipos, realiza los ejercicios que se piden

**data** Exp = Num N | Suma Exp Exp | Prod Exp Exp | Paren Exp **deriving** (Eq,Show)
**data** N = Dig D | ConsN N D **deriving** (Eq,Show)
**data** D = D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 **deriving** (Eq,Show)

**data** TreeD = NumT Int | SumaT TreeD TreeD | ProdT TreeD TreeD | ParenT TreeD **deriving** (Eq,Show)

1. Realiza la función **nToInt**, la cual a partir de un ***N***, regresa un Int.
   Firma de la función:
   nToInt :: N → Int
   Ejemplos:
   *Main> nToInt $ ConsN (ConsN (Dig D1) D0) D9
   109
   *Main> nToInt $ ConsN (Dig D2) D8
   28

2. Realiza la función **intToN**, la cual transforma un entero en un elemento del tipo *N*.
   Firma de la función:
   intToN :: Int → N
   Ejemplos:
   *Main> intToN 1024

ConsN (ConsN (ConsN (Dig D1) D0) D2) D4
*Main> intToN 543
ConsN (ConsN (Dig D5) D4) D3

3. Realiza la función **expToTree** que construye un árbol de derivación *TreeD*, a partir de una expresión de la gramática.
   Firma de la función:
   expToTree :: Exp → TreeD
   Ejemplos:
   *Main> expToTree $ Suma (Num (ConsN (ConsN (Dig D1) D0) D9)) (Num (ConsN (Dig D2) D8))
   SumaT (NumT 109) (NumT 28)
   *Main> expToTree $ Prod (Suma (Num (Dig D2)) (Num (Dig D4))) (Prod (Num (Dig D3)) (Num (Dig D2)))
   ProdT (SumaT (NumT 2) (NumT 4)) (ProdT (NumT 3) (NumT 2))

4. Define la función **treeToExp**, la cual construye una expresión de la gramática, a partir de su árbol de derivación.
   Firma de la función:
   treeToExp :: TreeD → Exp
   Ejemplos:
   *Main> treeToExp $ SumaT (SumaT (NumT 28) (NumT 46)) (ProdT (NumT 3) (NumT 2))
   Suma (Suma (Num (ConsN (Dig D2) D8)) (Num (ConsN (Dig D4) D6))) (Prod (Num (Dig D3)) (Num (Dig D2)))

   *Main> treeToExp $ ProdT (SumaT (NumT 2) (NumT 4)) (ProdT (NumT 3) (NumT 2023))
   Prod (Suma (Num (Dig D2)) (Num (Dig D4))) (Prod (Num (Dig D3)) (Num (ConsN (ConsN (ConsN (Dig D2) D0) D2) D3)))

5. Define la función **evalT**, la cual evaluará un arbol de derivación *TreeD*.
   Firma de la función:
   evalT :: TreeD → Int
   Ejemplos:
   *Main> evalT $ ProdT (SumaT (NumT 2) (NumT 4)) (ProdT (NumT 3) (NumT 2))
   36

*Main> evalT $ SumaT (NumT 2012) (NumT 23)
2035

6. Escribe las siguientes expresiones como elementos del tipo Exp

   *a*) 10*(258+47+369)
   *b*) 1000+((300*2)+10)
   *c*) 3*699 + 2
   *d*) (154+8016)*(299+734)
   *e*) (584+3115)*2
   *f*) 2 + 2

   Tomando de ejemplo el inciso **f**, deberás escribir en tu script:
   f = Suma (Num (Dig D2)) (Num (Dig D2))

7. Verifica que la evaluación de las expresiones que escribiste sean correctas, es dicir:

   *a*) 10*(258+47+369) = 6740
   *b*) 1000+((300*2)+10) = 1610
   *c*) 3*699 + 2 = 2099
   *d*) (154+8016)*(299+734) = 8439610
   *e*) (584+3115)*2 = 7398
   *f*) 2 + 2 = 4

   Siguiendo con el ejemplo del inciso f, deberás escribir:
   p6 = (evalT $ expToTreeE f) == 4
   En cuyo caso, si ejecutas desde terminal p6, deberá regresar True.

# Observaciones

- Les recomiendo empezar su proyecto lo más pronto posible.
- Los proyectos son **individuales**.
- Si requieren de cualquier función auxiliar para realizar su práctica, deben implementarlas ustedes.
- El asunto de la práctica es [**ED2018-1 Proyecto2**]
- Se enviará un correo automatico si el proyecto se envió con el asunto correcto

- Consulten los lineamientos de entrega antes de enviar
- Cualquier duda pueden mandarme correo

¡Suerte!