PLP - Primer Recuperatorio - 1er cuatrimestre de 2025

#Orden	Turno	Libreta	Apellido y Nombre	Ej1	Ej2	Ej3	Nota Final

Las notas para cada ejercicio son: -, I, R, B, MB. Aprueba obteniendo al menos dos ejercicios B y uno R, y se promociona con al menos dos ejercicios MB y uno B. Es posible obtener una aprobación condicional con un ejercicio MB, uno B y uno I. Esto requiere entregar algo que contribuya a la solución del ejercicio. Poner nombre, apellido y número de orden en todas las hojas, y numerarlas. Se puede utilizar todo lo definido en las prácticas y todo lo que se dio en clase, colocando referencias claras. El orden de los ejercicios es arbitrario. Recomendamos leer el parcial completo antes de empezar a resolverlo.

Ejercicio 1 - Programación funcional

Aclaración: en este ejercicio no está permitido utilizar recursión explícita, a menos que se indique lo contrario.

Se define el tipo de datos `LineaProd`, que representa líneas de producción de una fábrica. Una línea de producción se inicia con una lista de materiales, donde cada material se representa con un string. Tras esta primera etapa, se puede agregar un material a la línea, o se la puede unir con otra línea de producción.

```
data LineaProd = Materiales [String] | Agregar String LineaProd | Unir LineaProd LineaProd
```

Definimos las siguientes líneas para los ejemplos:

```
l1 = Unir
    (Materiales ["acero", "cobre"])
    (Unir (Agregar "madera" (Materiales ["madera"])) (Materiales ["mercurio"]))
l2 = Unir
    (Materiales ["acero", "cobre"])
    (Unir
        (Agregar "madera" (Materiales ["aluminio"]))
        (Agregar "aluminio" (Materiales ["plástico"])))
l3 = Unir
    (Materiales ["m1"])
    (Unir (Agregar "m3" (Materiales ["m3", "m4"])) (Materiales ["m1", "m2"]))
```

a) Dar los tipos y definir las funciones `foldLineaProd` y `recLineaProd`, que implementan respectivamente los esquemas de recursión estructural y primitiva para el tipo `LineaProd`. Solo en este inciso se permite usar recursión explícita.

b) Definir la función `materialesUsados :: LineaProd -> [String]`, que devuelve una lista con todos los materiales usados por una línea de producción. La lista devuelta no debe contener materiales repetidos. Por ejemplo:

`materialesUsados l1` ⇝ `["acero", "cobre", "madera", "mercurio"]`
`materialesUsados l2` ⇝ `["acero", "cobre", "madera", "aluminio", "plástico"]`

Sugerencia: usar `nub`.

c) Definir la función `sublineasDisjuntas :: LineaProd -> Bool`, que dada una línea de producción devuelve `True` si y solo si en ninguna etapa de la misma se unen dos líneas que usen al menos un material en común. Por ejemplo:

`sublineasDisjuntas l1` ⇝ `True`
`sublineasDisjuntas l2` ⇝ `False`, ya que dentro de ella hay dos sublíneas que se unen y tienen el material `"aluminio"` en común.
`sublineasDisjuntas l3` ⇝ `False`, ya que `l3` por sí misma es la unión de dos sublíneas que tienen a `"m1"` en común.

d) Definir la función `mismaEstructura :: LineaProd -> LineaProd -> Bool`, que dadas dos líneas de producción determina si son iguales sin considerar los materiales presentes en ellas. Por ejemplo:

`mismaEstructura l1 l2` ⇝ `False`
`mismaEstructura l1 l3` ⇝ `True`

Pista: aprovechar la currificación y utilizar evaluación parcial.

Ejercicio 2 - Demostración de propiedades

a) Considerar las siguientes definiciones:

```
data AIH a = Hoja a | Bin (AIH a) (AIH a)

      (++) :: [a] -> [a] -> [a]
{++0} [] ++ ys = ys
{++1} (x:xs) ++ ys = x : (xs ++ ys)

   reverse :: [a] -> [a]
{R} reverse = foldr (\ x rec -> rec ++ (x:[])) []

    foldr :: (a -> b) -> b -> [a] -> b
{F0} foldr f z [] = z
{F1} foldr f z (x:xs) = f x (foldr f z xs)
```

```
     hojas :: AIH a -> [a]
{H0} hojas (Hoja h) = [h]
{H1} hojas (Bin i d) = hojas i ++ hojas d

     espejo :: AIH a -> AIH a
{E0} espejo (Hoja h) = Hoja h
{E1} espejo (Bin i d) = Bin (espejo d) (espejo i)
```

Demostrar la siguiente propiedad:

$$\forall\ \texttt{x::AIH a .\ hojas (espejo x) = reverse (hojas x)}$$

Si se necesita un lema relacionado con `reverse` y `(++)`, no es necesario demostrarlo, pero sí enunciarlo, plantearlo como predicado unario y enunciar cada caso que se debería analizar para demostrarlo por inducción estructural, indicando la hipótesis inductiva para el o los caso(s) inductivo(s).

Se permite definir macros (i.e., poner nombres a expresiones largas para no tener que repetirlas).

No es obligatorio escribir los $\forall$ correspondientes en cada paso, pero es importante recordar que están presentes. Recordar también que los $=$ de las definiciones pueden leerse en ambos sentidos.

Se consideran demostradas todas las propiedades conocidas sobre enteros y booleanos.

b) Demostrar el siguiente teorema usando Deducción Natural, sin utilizar principios clásicos:

$$(\rho \Rightarrow \tau) \wedge (\sigma \Rightarrow \tau) \Rightarrow (\sigma \vee \rho) \Rightarrow \tau$$

Ejercicio 3 - Cálculo Lambda Tipado

Se desea extender el cálculo lambda simplemente tipado para modelar **regiones geométricas**. Para eso se extienden los tipos y expresiones de la siguiente manera:

$\tau ::= \cdots \mid \text{Región}$

$M ::= \cdots \mid \text{Rectángulo}(M, M) \mid \text{Unión}(M, M) \mid \text{foldRG}\, M \triangleright \text{Rectángulo}(x, y) \rightsquigarrow M; \text{Unión}(ri, rd) \rightsquigarrow M$

- Región es el tipo de las regiones del plano formadas por rectángulos y uniones de rectángulos.
- $\text{Rectángulo}(M_1, M_2)$ es un rectángulo de lados enteros no negativos, de ancho dado por M_1 y alto dado por M_2, cuyo extremo inferior izquierdo se encuentra en el cero del plano de coordenadas.
- $\text{Unión}(M_1, M_2)$ es la región del plano formada por la unión de las regiones M_1 y M_2.
- $\text{foldRG}\, M_1 \triangleright \text{Rectángulo}(x, y) \rightsquigarrow M_2; \text{Unión}(ri, rd) \rightsquigarrow M_3$ es el esquema de recursión estructural para regiones, donde las variables x e y se ligarán en M_2 al ancho y alto del rectángulo, mientrras que ri y rd se ligarán en M_3 a los resultados de la recursión sobre las regiones unidas.

a) Introducir las reglas de tipado para la extensión propuesta.

b) Definir el conjunto de valores y las nuevas reglas de reducción en un paso.

c) Mostrar paso por paso cómo reduce la expresión:

$$\text{foldRG Unión}((\lambda x : \text{Nat}.\text{Rectángulo}(\underline{1}, \underline{2}))\ \text{zero}, \text{Rectángulo}(\underline{3}, \underline{4})) \triangleright \text{Rectángulo}(x, y) \rightsquigarrow \text{True};\ \text{Unión}(ri, rd) \rightsquigarrow \text{if } ri \text{ then } rd \text{ else False}$$

d) Definir como macro la función contieneElPunto, que toma una Región y dos números naturales x e y, y devuelve True cuando el punto (x, y) está incluido en la región o en su borde, y False en caso contrario.

Asumir que se cuenta con la función $\leq : \text{Nat} \to \text{Nat} \to \text{Bool}$, la cual compara números naturales. Se pueden usar las macros **and** y **or** definidas en la práctica.