Autor: Leonardo Gallo

Lógica computacional
Facultad de Ciencias, UNAM

Práctica 1

Funciones y recursión

Fecha de entrega: antes de las 11:59pm del 27 de febrero de 2023.

Introducción:

El paradigma de la programación funcional tiene su base en el concepto de función, el cálculo-λ, la lógica combinatoria y las ecuaciones de recurrencia.

Haskell : es un lenguaje de programación puramente funcional (no posee ninguna característica imperativa); es perezoso (no realiza ningún cómputo que no sea estrictamente necesario); y es tipificado estáticamente (se conoce el tipo de cada expresión en tiempo de compilación). En Haskell podemos hacer uso de conceptos de alto nivel como las definiciones recursivas y los tipos de datos inductivos.

Entorno:

Usaremos GHC, Glasgow Haskell Compiler, el cual es un compilador de código libre para el lenguaje Haskell. Puedes descargarlo aquí, o instalarlo desde la línea de comando, por ejemplo:

- Ubuntu: `$ sudo apt-get install ghc`
- Fedora: `$ sudo dnf install ghc`

GHC puede cargar un script (usualmente con extensión *.hs*) y compilarlo, además, podemos usar el modo interactivo escribiendo:

```
$ ghci
```

Si queremos cargar un módulo a partir de un archivo *.hs* podemos ocupar la siguiente instrucción:

```
$ :load <archivo.hs>   -- podemos usar  la abreviatura :l
$ :load                -- retira de la memoria los módulos cargados.
$ :reload              -- recarga el último módulo cargado (abreviatura :r).
```

Guarda las siguientes líneas de código en un archivo llamada *MisFunciones.hs* y cargalo en el intérprete:

```
-- Módulo: MisFucniones
module MisFunciones where

constante x = 10

identidad :: Num a => a -> a
```

```
identidad x = x

cuadrado x = x * x

diferenciaPositiva x y = if x > y then x - y else y - x

factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Si todo sale bien aparecerá en pantalla la siguiente leyenda:

```
[1 of 1] Compiling MisFunciones ( MisFunciones.hs,
interpreted )
Ok, one module loaded
ghci>
```

Ahora, además de las operaciones predefinidas en Haskell, podrás usar tus funciones en el modo interactivo, para salir de **`ghci`** presiona **`Ctrl+d`**.

En Haskell los módulos empiezan con letra mayúscula y las funciones y variables con letra minúscula. Si queremos agregar un comentario de una línea usamos doble guión alto (`-- Coméntame`) y para comentarios de varias líneas usamos corchetes y guiones altos (`{- Coméntame -}`).

Objetivo:

Modelar soluciones a problemas matemáticos mediante un lenguaje puramente funcional.

Ejercicios:

Usando Haskell, implementa las siguientes funciones, en un archivo llamado *Practica01.hs*, (el valor de cada función es un punto):

1. **`esPalindromo`**, recibe como parámetro una lista y devuelve un booleano, **`True`** en caso de que la lista sea un palíndromo y **`False`** en el caso contrario.
2. **`divisores`**, recibe como parámetro un natural *n* y devuelve una lista con todos los divisores de *n*.
3. ***`primos`***, recibe como parámetro un natural *n* y devuelve una lista con todos los números primos entre 1 y *n*.
4. ***`sumaPares`***, recibe como parámetro una lista de pares ordenados y devuelve una par con el resultado de sumar todos los pares ordenados de la lista.
5. **`productoCartesiano`**, recibe como parámetros dos listas, *A* y *B,* y devuelve una lista con el producto cartesiano, *AxB*.
6. **`aplica`**, recibe como parámetros una función y una lista, y devuelve una lista con el resultado de evaluar cada elemento.

7. **`modulo`**, sin usar la función módulo predefinida en Haskell.
8. **`cadenaPar`**, recibe como parámetro una cadena y devuelve **`True`** si la longitud de la cadena es par y **`False`** en el caso contrario.
9. **`eliminaRepetidos`**, recibe como parámetro una lista y devuelve la lista sin repetidos.
10. **`ackermann`**, investigar definición.

Nota: Incluye comentarios sobre cada implementación.

Entrega:

Lineamientos para la entrega de prácticas de Lógica Computacional

1. Clonar el siguiente repositorio de laboratorio y aquí es donde se subirán sus entregas todo el semestre: https://github.com/ValeLanda/LogicaComputacional2023-2
2. Crear una carpeta con el nombre de sus equipos, dentro de esta carpeta deberán existir: un readme con los nombres y números de cuenta de los integrantes de equipo así como el número de práctica que se está resolviendo y el o los archivos que den solución a las prácticas.
3. Los equipos deberán ser de máximo 4 personas.
4. Las entregas deberán ser puntuales, es decir, deberán subirse antes de las 00:00 hrs del día establecido para la entrega.
5. Cualquier copia detectada será calificada con cero.
6. Para dudas o aclaraciones mandar correo a vale.landa@ciencias.unam.mx con el asunto: **[Logica2023-2]**