Facultad de Ciencias - UNAM
Lógica Computacional 2026-2

Práctica 0: Introducción a Haskell

Marco Vladimir Lemus Yañez
Fernando Cruz Pineda

Fecha de publicación: 10 02 2026
Fecha de entrega: 17 02 2026

1. Introducción

En esta práctica se trabajará con definiciones recursivas sobre números naturales y listas, así como con la definición de tipos algebraicos simples.

El objetivo es comprender cómo la recursión permite definir funciones sobre estructuras inductivas y cómo este mecanismo se relaciona con las definiciones inductivas utilizadas en matemáticas.

2. Objetivo

Que el estudiante sea capaz de:

- Implementar funciones recursivas sobre números naturales
- Implementar funciones recursivas sobre listas
- Definir un tipo algebraico simple
- Definir funciones recursivas sobre tipos definidos por el usuario

3. Justificación

La programación funcional permite modelar estructuras matemáticas mediante definiciones inductivas.

Comprender la relación entre recursión, tipos algebraicos e inducción estructural es fundamental para el diseño correcto de programas funcionales y para la comprensión formal de estructuras matemáticas.

4. Desarrollo de la Práctica

Todas las funciones deben implementarse utilizando recursión explícita y pattern matching.

No utilizar funciones predefinidas que resuelvan directamente el problema.

Parte A — Recursión sobre Números

Ejercicio 1. Fibonnaci

Definir una función que dado un natural n, regrese el n-ésimo elemento en la secuencia de fibonnaci

Ejemplos esperados:

```
fib 10 = 55
fib 7 = 13
```

Ejercicio 2. Módulo n

Definir una función que recibe dos naturales n,m y regresa m %n.

Ejemplos:

```
modulo 5 6 = 1
modulo 7 34 = 6
```

Ejercicio 3. MCD

Definir una función que calcule el máximo común divisor de dos números.

Ejemplos:

```
mcd 10 8 = 2
mcd 12 6 = 6
```

Parte B — Recursión sobre Listas

Ejercicio 4. Reversa de lista

Definir una función que calcule la reversa de una lista.

```
reversa [1,3,4,2] = [2,4,3,1]
```

Ejercicio 5. Máximo de elementos

Definir una función que calcule el máximo de una lista.

```
maximo [1,2,3,4] = 4
```

Ejercicio 6. Filtrar lista

Definir una función que reciba una lista l de enteros y regrese la lista con solamente los números pares de l.

```
pares [1,2,3,4,5] = [2,4]
pares [1,3,5,7] = []
```

Ejercicio 7. Contar ocurrencias

Definir una función que reciba un entero y una lista de enteros y regrese el número de veces que aparece el entero en la lista.

```
contar 2 [1,3,4,5,2] = 1
```

Parte C — Tipos Algebraicos

Ejercicio 8. Definición de números naturales

Definir el tipo:

```
data Nat = Z | S Nat
```

Ejercicio 9. Conversión a entero

Definir una función que convierta un valor de tipo `Nat` a entero.

```
convertir (S (S (S Z))) = 3
convertir Z = 0
```

Ejercicio 10. Multiplicación sobre Nat

Definir la multiplicación de dos números naturales definidos con el tipo anterior, no utilizar la función anterior y realizar la suma definida en Haskell.

```
mult (S (S (S Z))) (S Z) = S(S(S(Z)))
```

5. Especificaciones de Entrega

- Entregar un archivo `.hs`
- El archivo debe compilar y ejecutar sin errores
- Incluir comentarios explicando cada función
- No usar funciones de biblioteca que resuelvan directamente los ejercicios

Formato del archivo
El archivo debe incluir:

- Nombre del alumno
- Fecha
- Descripción breve de cada función

Forma de entrega
Enviar el archivo por correo electrónico.