1. Tipos básicos (`Bool` , `Char` , `Int` , `Integer` , `Float`)

1. Indicá el **tipo** de cada una de estas expresiones:
   - `True && False`
   - `'a' == 'b'`
   - `5 + 3`
   - `5 / 2`
   - `fromIntegral (length "hola")`
2. Decidí cuáles son **válidas** y cuáles no, y explicá por qué:
   - `'a' + 1`
   - `not 3`
   - `3 == 3.0`
   - `5 < 10`

---

2. Tipo Lista

3. Indicá el tipo de cada lista:
   - `[1,2,3]`
   - `['a','b','c']`
   - `[[1,2],[3,4]]`
   - `[True, False, True]`
4. Escribí funciones que:
   - Devuelvan el **primer elemento** de una lista no vacía.
   - Devuelvan `True` si una lista está vacía y `False` en caso contrario.
   - Cuenten cuántos elementos tiene una lista (sin usar `length`).

---

3. Tipo Tupla

5. Indicá el tipo de estas expresiones:
   - `(1, 'a')`
   - `(True, 3, "hola")`
   - `((1,2), (3,4))`
6. Definí funciones que:
   - Dado un par `(x,y)` devuelva `y` .
   - Dada una tupla `(Int, Int)` , devuelva la suma de sus componentes.

---

4. Tipo Función

7. Indicá el tipo de las siguientes funciones (sin usar GHCi):
   - `f x = x + 1`
   - `g x y = x == y`

- `h xs = head xs`

8. Pensá un ejemplo de una función cuyo tipo sea:
   - `Int -> Bool`
   - `Char -> Int`
   - `[a] -> Bool`

---

5. Funciones currificadas

9. Dada la función:

```
suma x y = x + y
```

   - ¿Cuál es su tipo?
   - ¿Qué tipo tiene `suma 3` ?

10. Definí una función que:
- Reciba un número
- Devuelva una función que sume ese número a otro

---

6. Tipos polimórficos (`a`)

11. Indicá cuáles de estas funciones son polimórficas:
- `id x = x`
- `f x = x + 1`
- `g xs = head xs`

12. Escribí una función polimórfica que:
- Reciba dos valores y devuelva el primero.

---

7. Tipos sobrecargados (`+` , `==` , etc.)

13. Explicá qué significa este tipo:

```
(+) :: Num a => a -> a -> a
```

14. Indicá por qué estas funciones usan **sobrecarga**:
- `sum`
- `(*)`
- `(==)`

---

8. Expresiones condicionales (`if then else`)

15. Definí una función que:

- Reciba un número
- Devuelva `"positivo"` o `"negativo"`

16. ¿Por qué en Haskell el `else` es obligatorio?

---

9. Pattern Matching

17. Definí una función que:
- Devuelva `True` si un `Bool` es `True`
- Y `False` en el otro caso (usando pattern matching).

18. Reescribí con pattern matching:

```
f x = if x == 0 then True else False
```

---

10. Patrones de Tuplas

19. Definí una función que:
- Reciba una tupla `(x,y)`
- Devuelva `True` si `x > y`

20. Definí una función que:
- Reciba una tupla `(Int, Int, Int)`
- Devuelva el mayor de los tres números.

---

11. Patrones de Listas

21. Definí funciones usando pattern matching que:
- Devuelvan el primer elemento de una lista.
- Devuelvan la lista sin su primer elemento.
- Devuelvan `True` si la lista tiene un solo elemento.

22. Definí una función que:
- Sume todos los elementos de una lista de enteros.