

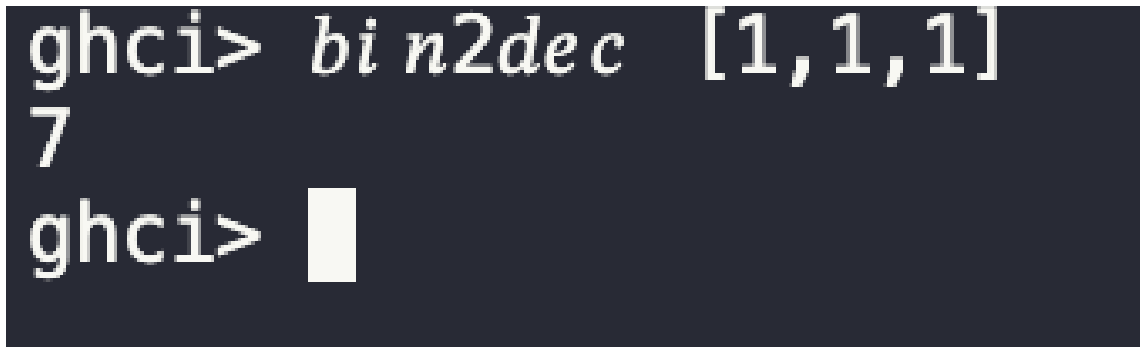
# Capturas de teste das funções

Henrique Zapella Rocha\*  
Programação Funcional — PUCRS

26 de abril de 2024

## Capturas das funções criadas funcionando

1. Definir uma função recursiva que recebe um número binário (interpretado como número inteiro sem sinal) e retorna o valor equivalente em decimal. `bin2dec :: [Int] -> Int`



```
ghci> bin2dec [1,1,1]
7
ghci> 
```

**Figura 1:** Primeira função

2. Definir uma função recursiva que recebe um número decimal inteiro não-negativo, um número de bits desejado e retorna o valor equivalente em binário (interpretado como número inteiro sem sinal) com o número de bits informado. Caso o número de bits não seja suficiente, escolha uma forma adequada de lidar com a situação, por exemplo, usando truncamento ou mensagem de erro. Por exemplo, `Dec2bin 2 8` deve retornar `[0,0,0,0,0,0,1,0]`. `dec2bin :: Int -> Int -> [Int]`



```
ghci> dec2bin 2 8
[0,0,0,0,0,0,1,0]
ghci> 
```

**Figura 2:** Segunda função

---

\*henrique.z@edu.pucrs.br

3. Definir uma função recursiva que recebe um número binário na representação de complemento de dois e retorna o valor equivalente em decimal inteiro. `bincompl2dec :: [Int] -> Int`

```
ghci> bincompl2dec [1,0,0,0,1]
-15
ghci> █
```

**Figura 3:** Terceira função

4. Definir uma função recursiva que recebe um número decimal inteiro, um número de bits desejado e retorna o valor equivalente em binário na representação de complemento de dois com o número de bits informado. Caso o número de bits não seja suficiente, escolha uma forma adequada de lidar com a situação, por exemplo, usando truncamento ou mensagem de erro. Por exemplo, `dec2bincompl (2) 8` deve retornar `[1,1,1,1,1,1,1,0]`. `dec2bincompl :: Int -> Int -> [Int]`

```
ghci> dec2bincompl (-2) 8
[1,1,1,1,1,1,1,0]
ghci> █
```

**Figura 4:** Quarta função

5. Definir uma função recursiva que recebe dois números binários na representação de complemento de dois e retorna a soma binária destes dois valores dentro do limite de número de bits suportado indicado pelo terceiro parâmetro. Assuma que os números a serem somados e a resposta possuam exatamente o número de bits indicado no terceiro parâmetro. A função não implementa nenhum tipo de notificação para casos de overflow/underflow. `somarbin :: [Int] -> [Int] -> Int -> [Int]`

```
ghci> somarbin [1,1,1,1] [1,1,1,1] 4
[1,1,1,0]
ghci> █
```

**Figura 5:** Quinta função caso que é permitido

```
ghci> somarbin [1,1,1] [1,1,1] 4
*** Exception: Número de bits insuficiente para representar o valor.
CallStack (from HasCallStack):
```

**Figura 6:** Quinta função caso que não é permitido