Paradigmas de Programação - Lista 5
Giovani Zanella e Sofia Effting

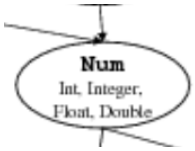
1) O resultado será (False, -4). Como 1 não é igual a 4, retorna-se False. A conjunção de falso com verdadeiro também resulta em falso. A função mod calcula o resto da divisão de 32 (4 * 8) por 31. O resto, que é 1, é então elevado ao quadrado e é subtraído 5, resultando em -4.

2) O ^ é um operador para o tipo inteiro e o ** é um operador para o tipo float:
   a) Prelude> 2^0.5
      error
   b) Prelude> 2**0.5
      1.414213562

3) dobro :: Double -> Double
   dobro x = x + x

4) incremento x = x + 1
   decremento x = x - 1

5) Acho melhor explicar cada parte separada.
   O comando :t é usado para ver os tipos de uma função, nesse caso queremos o tipo da expressão `decremento (incremento 9)`. O comando primeiro executa a função incremento para o nove, depois faz o decremento para o resultado que sai da função incremento que no caso seria 10.
   Já a parte do Num a => a, o resultado é um número que tem uma restrição a classe Num, ou seja
   O resultado pode ser qualquer tipo que pertença a classe num.



6) sobeDesce :: (Num t, Num t) => (t, t) -> (t, t)
   sobeDesce (x, y) = (x+1, y-1)

7) incremento x = x + 1
   decremento x = x - 1
   sobeDesce2 (x, y) = (incremento x, decremento y)

8) trocaValor (a, b) = (b, a)

9) Prelude> negate (-1)
   1

10) O erro do programador foi não utilizar parênteses. O código dele é: negate -8, o certo seria: negate (-8). O problema por trás disso é igual ao visto em sala com o -9, a função negate recebe um argumento, porém estamos passando -8 e o “-” também é uma função que recebe um argumento. Pelo meu entendimento e o que foi dito em sala, ao chamar a função negate -8, o compilador Haskell tenta aplicar a função negate à função -, o que causa o erro.