Lista Haskell

1) Resultado (**False**, **-4**).

A vírgula em (1 == 4 && True, mod (4*8) 31^2-5) faz com que o programa interprete como se fosse duas expressões separadas. Na primeira parte (1 == 4 && True) o programa faz a comparação entre 1 e 4 que é False, depois disso (False && True) e retorna **False** como a primeira parte do resultado. Na segunda parte (mod (4*8) 31^2-5) o programa pede o resto da divisão entre (4*8) e 31^2 ou seja, o resto da divisão entre 961 e 32 que resulta em 1. Em seguida o programa subtrai 5 de 1 retornando **-4** como o segundo resultado.

---

2) Qualquer valor dado retorna tipos diferentes nas duas funções.
Além disso, o operador "^" aceita apenas números naturais.

---

3) função dobrar argumento.

```
dobro :: Double -> Double
dobro a = a * 2
main = do
        print(dobro 0+2) --teste valor positivo e negativo
        print(dobro 0-2)
```

---

4) Função incremento e decremento

```
incremento :: Double -> Double
incremento a = a + 1

decremento:: Double -> Double
decremento a = a - 1

main = do
        print(incremento 0+1) --teste valor positivo e negativo
        print(decremento 0-1)
```

---

5) incremento e decremento representam duas funções isoladas.
:t representa o tipo da expressão.
O resultado da expressão é do tipo a (inferência de tipos feita estaticamente). Restrição: o tipo a deve ser numérico.

---

6) Função sobe-desce, recebe um ponto e devolve um ponto (ordenado)

usando definição matemática de par ordenado da wikipedia:
"For any two objects a and b, the ordered pair (a, b) is a notation specifying the two objects a and b, in that order"

```
sobeDesce :: (Num a, Num b) => (a, b) -> (a, b)
sobeDesce (a, b) = (a+1, b-1)

main = do
        print(sobeDesce(0-1,0-1)) --teste valor positivo e negativo
```

---

7)Implemente a função sobeDesce2, semelhante à questão anterior,
mas utilizando as funções incremento e decremento.

```
incremento :: Int -> Int
incremento a = a + 1

decremento :: Int -> Int
decremento a = a - 1

sD2 :: (Int, Int) -> (Int, Int)
sD2 (a, b) = ( incremento(a), decremento(b) )

main = do
        print( sD2(0, 0) )
```

---

8) função que inverte as posições de a por b no ponto ordenado.

```
trocaValor :: (Num a, Num b) => (a, b) -> (b, a)
trocaValor (a, b) = (b, a)

main = do
        print(trocaValor( 11, 0-5) ) --teste valor positivo e negativo
```

---

9) função que nega um número.

```
negar :: Num a => a -> a
negar a = 0 - a

main = do
```

```
print( negar(0 -1) ) -- teste valor negativo
print( negar(0 +1) ) -- teste valor positivo
```

---

10) habilitar contexto flexível como o compilador sugere.