Programação Funcional
Lista de Exercícios 01

Prof. Wladimir Araújo Tavares

1. Qual é o tipo mais geral para a definição seguinte?

```
h x y = if x>=y then x-y else y-x
```

   (a) `h :: (Ord a, Eq a) => a -> a -> a`
   (b) `h :: (Num a, Ord a) => a -> a -> a`
   (c) `h :: (Num a, Eq a) => a -> a -> a`

2. Qual o resultado da expressão `filter ( \ x-> mod x 2 ==0) [0..9]` ?

   (a) `[1,3,5,7,9]`
   (b) `[0,2,4,6,8]`
   (c) `[0]`

3. Qual é o resultado da expressão `length [0,2..6]` ?

   (a) 4
   (b) 5
   (c) 6

4. Qual dos tipos é admissível para a definição seguinte?

```
f xs ys = sum [x*y | (x,y)<-zip xs ys]
```

   (a) `f :: ([Float],[Float]) -> Float`
   (b) `f :: [Float] -> [Float] -> [Float]`
   (c) `f :: [Float] -> [Float] -> Float`

5. Qual o resultado da expressão `take 4 (drop 3 [1..])` ?

   (a) `[4,5,6,7]`
   (b) `[3,4,5,6]`
   (c) `[5,6,7,8]`

6. Qual das seguintes atribuições de tipos é correta?

   (a) `('a',[2,3]) :: (Char,[Int])`
   (b) `('a',[2,3]) :: [(Char,Int)]`
   (c) `('a',[2,3]) :: ([Char],[Int])`

7. Qual é o resultado da expressão `[1]:[]:[2]:[3]:[]` ?

   (a) um erro de tipos
   (b) `[[1],[2],[3]]`
   (c) `[[1],[],[2],[3]]`

8. Considere a definição seguinte:

```
g [] = []
g (x:_:xs) = x : g xs
```

   Qual o resultado de `g "abdef"` ?

   (a) "abde"
   (b) "ae"
   (c) um erro de execução

9. Qual das seguintes propriedades é verdadeira para todas as listas xs, ys e funções f?

   (a) `map f (xs++ys) = map f (ys++xs)`
   (b) `map f (xs++ys) = map f ys ++ map f xs`
   (c) `map f (xs++ys) = map f xs ++ map f ys`

10. Qual o resultado da expressão `zip [1..] "abc"` ?

    (a) é uma lista finita
    (b) é uma lista infinita
    (c) um erro de execução

11. Qual é o tipo mais geral para a função maximum cujo resultado é o maior valor duma lista?

    (a) `maximum :: Ord a => [a] -> a`
    (b) `maximum :: Num a => [a] -> a`
    (c) `maximum :: [a] -> a`

12. A saída de `length [x+y | x <- [1..10], y <- [1..5], x>y]` deve ser

    (a) 100
    (b) 25
    (c) 70
    (d) 35
    (e) 50

13. Qual dos seguintes tipos é admissível para a função `f xs = reverse xs == xs` ?

    (a) `f :: [Int] -> Bool`
    (b) `f :: [Int] -> Int`
    (c) `f :: [Bool] -> [Bool]`

14. Qual é o tipo mais geral admissível para a função elem do prelúdio-padrão que verifica se um valor ocorre numa lista?

    (a) `elem :: a -> [a] -> Bool`
    (b) `elem :: Ord a => a -> [a] -> Bool`
    (c) `elem :: Eq a => a -> [a] -> Bool`

15. Seja a função haskell,

```
teste u n = [sum [y+n | y <- (take w u)] | w <- [1..length u]]
```

    Então teste [1,2,3] 3 retorna,

    (a) `[3,8,14]`
    (b) `[2,12,45]`
    (c) `[1,6,12]`
    (d) `[0,10,25]`
    (e) `[4,9,15]`

16. Seja a definição de função haskell,

```
gamma [] = ""
gamma [c] = [c]
gamma (c:u) = c:'.': gamma u
```

    Marque a proposição INCORRETA,

17. `gamma "casa"==> "c.a.s.a"`
18. `gamma $ '.':[] ==> "."`
19. `gamma ['a', 'b', 'c'] ==> "a.b.c"`
20. `gamma ['c', '.'] ==> "c."`
21. `gamma ==>`
22. Indique o tipo mais geral para as seguintes definições; tenha o cuidado de incluir restrições de classes no caso de operações com sobrecarga.

    (a) `segundo xs = head (tail xs)`
    (b) `trocar (x, y) = (y, x)`
    (c) `par x y = (x, y)`
    (d) `dobro x = 2*x`
    (e) `metade x = x/2`
    (f) `minuscula x = x >= 'a' && x <= 'z'`
    (g) `intervalo x a b = x >= a && x <= b`
    (h) `palindromo xs = reverse xs == xs`
    (i) `twice f x = f (f x)`

23. Mostre que a função last (que seleciona o último elemento de uma lista) pode ser escrita como composição das funções do prelúdio-padrão apresentadas na primeira aula. Consegue encontrar duas definições diferentes?

    Dica: Use as funções `reverse, head, take, (!!), length`.

24. Num triângulo, verifica-se sempre a seguinte condição: a medida de um qualquer lado é menor que a soma da dos outros dois. Complete a definição de uma função `triangulo a b c = ...` que testa esta condição, o resultado deve ser um valor boleano.
25. Construa a função `inserir`:
    INPUT: Número x e lista u de números ordenados ascendentemente
    OUTPUT: Lista de números ordenados ascendentemente oriunda da inserção apropriada de x em u
    `inserir 3 [2,7,12] ==> [2,3,7,12]`
26. Considere uma função `count :: (a -> Bool) -> [a] -> Int` que conta o número de elementos de uma lista para os quais o predicado dado é True. Exemplos:

```
> count (>2) [0,1,2,3]
1
> count (/='a') "banana"
3
```

    (a) Escreva uma definição recursiva da função count.
    (b) Escreva uma definição não-recursiva de count usando filter e length.