Departamento de Matemática da Universidade de Coimbra		
2008/2009	**Programação Funcional**	**Folha 1** (v1.5)

1. Sabendo que:

   ```
   odd :: Integer -> Bool
   not :: Bool -> Bool
   round :: Double -> Integer
   ```

   indique (caso exista) o tipo de cada um dos seguintes termos:

   (a) `(odd 3,[not True, False])`

   (b) `not (round 3.0)`

   (c) `not(odd (round (0.5 + 3.5)))`

   (d) `[(3.5, round, odd)]`

2. Indique um termo de cada um dos seguintes tipos:

   (a) `[(Int,Int)]`

   (b) `([Int],Int)`

   (c) `[Int -> Bool]`

   (d) `(Int,(Bool,Int))`

3. Escreva a definição de uma função que:

   (a) dado um par de números inteiros, dê como resultado o maior desses números. Qual o tipo desta função?

   (b) que dados 3 números, dê como resultado o maior dos três.

   (c) Reescreva a definição anterior sem usar "ifs" (i.e., usando a função definida em 3a).

   (d) Repita as duas alíneas anteriores para a função `max4` que recebe como parâmetros 4 números e retorna o maior deles.

4. Defina duas funções para calcular, respectivamente, a área interior e o perímetro de uma circunferência, dado o seu raio.

5. Um número diz-se perfeito quando é igual à soma de todos os seus divisores (incluindo o 1 mas excluindo-se a si próprio). Por exemplo $6 = 1+2+3$ é um número perfeito. Escreva um predicado que teste se um dado número é "perfeito".

6. Num triângulo verifica-se sempre que o comprimento de qualquer lado é sempre inferior (ou igual) à soma dos comprimentos dos outros dois; esta propriedade designa-se por *desigualdade triangular*.

   (a) Escreva a definição de uma função `tri` que dados três números, correspondentes aos comprimentos dos lados de um triângulo, dê como resultado verdadeiro se esses números verificam a desigualdade triangular.

   (b) Escreva uma definição alternativa da função `tri` que tome partido do facto de que basta testar se o maior dos números é menor (ou igual) à soma dos outros dois.

7. Defina a função `productoLista` que, dado uma lista de inteiros, calcule o produto de todos os seus elementos.

8. Considere uma lista `l` e um elemento `elem`, escreva definições de funções para:

(a) determinar se a lista está vazia;

(b) determinar se o elemento pertence à lista;

(c) calcular o comprimento da lista;

(d) calcular o número de ocorrências do elemento na lista.

9. Defina uma função, `fusão`, que faça a fusão de duas listas (de inteiros) ordenadas, numa lista ordenada.

10. Defina uma função, `troca`, que tome como argumento uma lista de pares e devolva a mesma lista de pares mas com os elementos dos pares trocados entre si.

11. Defina uma função, `seginicial`, que verifique se uma lista é, ou não, um segmento inicial de uma outro lista. Uma lista é um segmento inicial de ela própria, e a lista vazia é um segmento inicial de uma qualquer lista.

12. Defina uma função que, dado um conjunto A (representado por uma lista) calcule o conjunto das partes do conjunto A (conjunto de todos os sub-conjuntos de A).

13. Defina uma função, `posição`, que dê a posição de uma "substring" dentro de uma outra "string". Por exemplo `posição("end","extend")` devolve o par (4,6) ((princípio,fim)). Se a "substring" não ocorrer deve-se providenciar uma situação de erro. Qual é a posição "razoável" para a "substring" vazia?

14. Escreva funções que, dada uma relação binária (representada por uma lista de pares), determine:

    (a) O seu fecho reflexivo;

    (b) O seu fecho transitivo;

    (c) O seu fecho simétrico.

15. Utilizando as funções definidas no exercício anterior, escreva um predicado que verifique se uma dada relação é uma relação de equivalência.

16. Defina uma função que receba como argumento uma lista e uma relação de ordem (total) apropriada, e devolva a lista ordenada correspondente.

17. Defina uma função, `map2`, que aplique uma função a todos os elementos de todas as listas numa lista de listas.

18. Considere as seguintes definições:

```
fact :: Int -> Int
fact n = n*fact(n-1)
fact1 :: Integer -> Integer
fact1 n = n*fact1(n-1)
```

    (a) Qual o resultado de `fact 20`? Porquê?

    (b) qual o resultado de `fact1 (fact 3)`? Porquê?

    (c) Re-escreva a função `fact` de forma a usar acumuladores. Compare a eficiência das duas definições.

19. Defina uma função que calcule, para um dado número de discos, a solução do problema das Torres de Hanói.