Engenharia de Software
Campus Garanhuns

Disciplina: Programação 3 - Programação Funcional
Professor: Emanoel Barreiros
Assunto: Funções recursivas, funções de alta ordem e compreensão de listas.

Resumo: Utilize a linguagem de programação Haskell para resolver os problemas desta lista. Esta lista aborda o conceito de funções recursivas, funções de alta ordem e compreensão de listas.

1) Implemente uma função que receba uma lista de inteiros (que pode ou não estar ordenada) e retorne uma lista ordenada em ordem crescente formada apenas pelos números ímpares da lista recebida.
   Exemplo: impares [3,6,4,8,1,9,7]
   Saída: [1,3,7,9]

2) Defina uma função que retorne o elemento na n-ésima posição de uma lista.
   Exemplo: posicao 2 [‘a’, ‘b’, ‘c’, ‘d’]
   Saída: ‘c’

3) Defina uma função que repita as ocorrências até um determinado valor, no formato de uma lista, tal que (NÃO PODE USAR O *replicate*):
   Exemplo: repete 4
   Saída 1: [[4,4,4,4], [3,3,3], [2,2], [1]]
   Saída 2: [4,4,4,4,3,3,3,2,2,1]

4) Construa uma função que cheque se o conteúdo de uma lista é um palíndromo:
   palindromo [1,2,3,4,5] = False
   palindromo [1,2,3,2,1] = True
   palindromo [1,2,2,1] = True

5) Construa uma função que retorne os n primeiros elementos da sequência de Fibonacci:
   Exemplo: fibonacci 10
   Saída: [0,1,1,2,3,5,8,13,21,34]

6) Sem olhar as definições no Prelude, defina a seguintes funções de alta ordem:
   a) Decide se todos os elementos de uma lista satisfazem um predicado:
      `all :: (a -> Bool) -> [a] -> Bool`
   b) Decide se algum elemento de uma lista satisfaz um predicado:
      `any :: (a -> Bool) -> [a] -> Bool`
   c) Selecione elementos de uma lista enquanto eles satisfazem um predicado:
      `takeWhile :: (a -> Bool) -> [a] -> [a]`
   d) Remove elementos de uma lista enquanto eles satisfazem um predicado:

```
dropWhile :: (a -> Bool) -> [a] -> [a]
```

7) Redefina as funções `map` e `filter` usando `foldr`.

8) Usando foldl, defina a função dec2int :: [Int] -> Int que converte uma lista de inteiros em um inteiro.

   Exemplo: dec2int [2,3,4,5] deve retornar 2345

9) Considere a função unfold que encapsula o padrão recursivo definido abaixo

```
unfold p h t x
        | p x = []
        | otherwise = h x : unfold p h t (t x)
```

   Isto é, a função `unfold` produz uma lista vazia se o predicado é verdadeiro para o argumento passado em `x`, caso contrário, produz uma lista não vazia aplicando `h` a `x`, para formar a cabeça, e a função `t` aplicada a `x` que é processado recursivamente usando as mesmas regras, produzindo a cauda da lista. Como exemplo, podemos definir uma função `int2bin`, que converte um número inteiro em uma lista de bits:

```
int2bin = unfold (== 0) (`mod` 2) (`div` 2)
```

   Redefina as funções `map f` e `iterate f` da biblioteca padrão usando a função `unfold`.

10) Defina a função `altMap :: (a -> b) -> (a -> b) -> [a] -> [b]` que aplica de forma alternada as duas funções que recebe como argumento a elementos sucessivos em uma lista.

    Exemplo: altMap (+10) (+100) [0,1,2,3,4] deve retornar [10, 101, 12, 103, 14]

11) Sem olhar nas definições do Prelude, defina uma função de alta ordem chamada curry que converte uma função em um par (tupla) em uma versão currificada. Defina também uma função chamada uncurry que converte uma função currificada para dois argumentos em uma função que recebe um par (tupla).