

EXAME - 1ª Chamada
23.Janeiro.2003
Duração: 2:30 horas

Paradigmas da Programação I
LMCC + LESI

NOME: _____

CURSO: _____ NUM: _____

I

Considere a codificação de inteiros positivos em base 2. Em HASKELL a representação binária pode ser efectuada através de uma lista de valores booleanos (com o *bit* menos significativo à esquerda) usando as seguintes declarações.

```
type Bits = [Bool]
```

```
bits :: Int -> Bits
```

```
bits n = if n > 0 then (r == 1) : (bits q)  else []  
        where (q,r) = n `divMod` 2
```

1. Escreva a função `bitsInt :: Bits -> Int`, inversa da anterior, que descodifica uma sequência de *bits* no inteiro correspondente.

2. Escreva uma função `&& :: Bits -> Bits -> Bits` que, dadas duas representações binárias, calcula uma terceira representação binária formada pelo e lógico (bit a bit) dos argumentos.

3. Escreva a função `_e_ :: (Int,Int) -> Int` que toma como argumento um par de inteiros e dá como resultado o inteiro cuja representação binária é o e lógico (bit a bit) das representações binárias dos elementos do par.



NOME: _____

CURSO: _____ NUM: _____

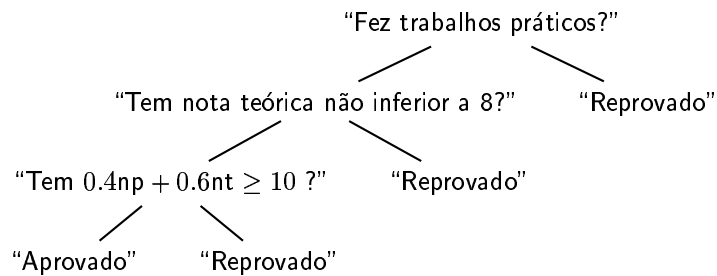
II

Considere que, para armazenar um questionário cujas respostas às questões são sempre da forma sim/não, se definiu o seguinte tipo de dados:

```
data Questionario a = Resultado a
                    | Questao String (Questionario a) (Questionario a)
```

Assume-se que uma resposta afirmativa (negativa) à questão implica continuar com o questionário da subárvore esquerda (direita).

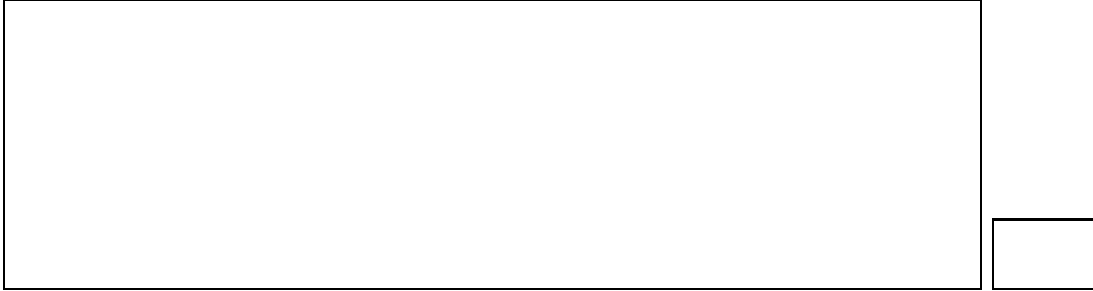
Um exemplo de um pequeno questionário é a árvore



1. Apresente o termo HASKELL a que corresponde a árvore apresentada no exemplo. Qual o seu tipo ?

2. Defina uma função `resp :: [Bool] -> Questionario a -> Maybe a` que dada uma sequência de respostas e um questionário calcula o eventual resultado.

3. Defina a função `questiona :: Show a => Questionario a -> IO ()` que dado um questionário vai, de modo interativo, colocando as questões ao utilizador e, no final, apresenta o resultado encontrado.



NOME: _____

CURSO: _____ NUM: _____

III

Considere a seguinte definição em Haskell:

```
bubble :: (Ord a) => [a] -> [a]
bubble [] = []
bubble [x] = [x]
bubble (a:b:bb) = let (c:cc) = bubble (b:bb)
                  in if a < b then a:c:cc
                     else c:a:cc
```

1. Explique porque é que a função tem esse tipo (e não apenas `[a] -> [a]`).

2. Calcule o valor de `bubble [9,11,10,8]`.

3. Use a função `bubble` para definir uma função que ordena uma lista.