BCC222 — Programação Funcional
Segunda prova 2014-1
Matrícula:
Nome:

DECOM UFOP 04 de junho de 2014
Prof. José Romildo Malaquias

---

1 Instruções

- Não se esqueça de preencher o nome e o número de matrícula na folha de respostas.
- A avaliação é individual.
- É permitida a consulta ao material das aulas.
- A interpretação das questões faz parte da avaliação.
- Faça as observações que achar necessárias por escrito.
- Os computadores do laboratório podem ser usados para resolver as questões de programação. Porém o **cabo de rede deve estar desconectado** após a autenticação do usuário.
- **Não é permitido o uso de notebook** ou outro dispositivo com acesso à rede sem fio.
- Caso tenha alguma dúvida sobre o tipo de funções e variáveis da bilbioteca de Haskell, use o comando `:type` do ghci para consultá-lo. O comando `:browse` pode ser usado para listar os nomes definidos em um módulo. O comando `:info` pode ser usado para exibir informações mais completas sobre um nome.
- Ao final da prova o aplicador copiará os arquivos contendo suas respostas para um disponsitivo de armazenamento externo. Organize os seus arquivos em um **diretório com o seu nome**.
- **Não use espaços e nem caracteres especiais ou com acento nos nomes do diretório e dos arquivos**
- Use um arquivo para cada questão da prova.
- Qualquer tentativa de cola será devidamente punida.

Todas as definições solicitadas devem ser acompanhadas de uma assinatura de tipo.

2 Questões

Questão 1. [3 PONTOS]
Defina uma ação de entrada e saída que, quando executada, lê um número, digamos n, e então lê outros n números e finalmente retorna a média aritmética dos n números lidos.

Questão 2. [3 PONTOS]
Faça um programa que leia o salário inicial de um funcionário e calcule e exiba o salário a receber, dado pelo salário inicial acrescido de bonificação e de auxílio escola, de acordo com as tabelas dadas.

salário inicial	**bonificação**
até R$500,00	12% do salário inicial
entre R$500,00 e R$1.200,00	5% do salário inicial
acima R$1.200,00	sem bonificação

salário inicial	**auxílio escola**
até R$600,00	R$150,00
acima R$600,00	R$100,00

Questão 3. [4 PONTOS]
Considere a seguinte aplicação para jogar o jogo ***adivinhe o número***, como explicado a seguir.

1. O programa escolhe um número a ser adivinhado pelo jogador (usuário) selecionando um número inteiro aleatório no intervalo de p a q, onde p e q são dois números naturais informados como argumentos da linha de comando.

2. O programa exibe a mensagem `Adivinhe um número entre p e q`.

3. O jogador informa o seu palpite.
4. Se o palpite do jogador estiver incorreto:
   - o programa exibe a mensagem *Muito alto* ou *Muito baixo* convenientemente para ajudar o jogador a acertar o número nas próximas jogadas.
   - o jogo continua com o programa solicitando o próximo palpite e analisando a resposta do usuário.
5. Quando o jogador insere a resposta correta o programa exibe a mensagem *Parabéns, você adivinhou o número*

```haskell
module Main (main) where

import System.IO ( stdout, hSetBuffering, BufferMode(NoBuffering) )
import System.Random ( randomRIO )
import System.Environment ( getArgs, getProgName )
import System.Exit ( exitFailure )

main :: IO ()
main =
  do hSetBuffering stdout NoBuffering
     putStrLn "Adivinhe o número v1.0"
     putStrLn "============================================"
     args <- getArgs
     case args of
       [arg1,arg2] -> do let p = read arg1
                         let q = read arg2
                         numero <- randomRIO (p,q)
                         acertar p q numero
       _ -> do prog <- getProgName
               putStrLn "Chamada incorreta do programa"
               putStrLn ("Uso: " ++ prog ++ " <limite inferior> <limite superior>")
               exitFailure

acertar :: Int -> Int -> Int -> IO ()
acertar p q numero =
  do putStrLn ""
     putStr ("Adivinhe um número entre " ++ show p ++ " e " ++ show q ++ ": ")
     x <- readLn
     if x == numero
       then putStr "Parabéns, você adivinhou o número"
       else do putStr (if x < numero then "Muito baixo." else "Muito alto.")
               putStrLn " Tente novamente"
               acertar p q numero
```

Exemplo de execução da aplicação

```
$ ./adivinha 3 17
Adivinhe o número v1.0
========================================

Adivinhe um número entre 3 e 17: 6
Muito baixo. Tente novamente

Adivinhe um número entre 3 e 17: 10
Muito baixo. Tente novamente

Adivinhe um número entre 3 e 17: 15
Muito alto. Tente novamente

Adivinhe um número entre 3 e 17: 12
Muito baixo. Tente novamente

Adivinhe um número entre 3 e 17: 14
Muito alto. Tente novamente

Adivinhe um número entre 3 e 17: 13
Parabéns, você adivinhou o número
```

Reescreva este programa de forma que o usuário tenha um número máximo de tentativas para adivinhar o número. Para tanto o programa deverá receber um terceiro argumento na linha de comando para indicar o número máximo de tentativas. O usuário vence o jogo se ele adivinhar o número dentro do número máximo de tentativas. Caso ele perca o jogo, deverá ser exibida uma mensagem encorajando-o a tentar novamente.