UNIVERSIDADE FEDERAL DO PARANÁ – CENTRO POLITÉCNICO

Disciplina: CI1062 – Paradigmas de Programação / CI062 - Técnicas Alternativas de Programação

Professora: Rachel Reis

EXERCÍCIOS EM HASKELL
AULA PRÁTICA EM LABORATÓRIO

1) Crie um projeto no Replit e complete o módulo Main conforme mostrado abaixo. Em seguida, teste as funções de saída (1) putStrLn, (2) putStr, (3) put e observe a diferença entre eles.

```
module Main where

main :: IO()
main = do
```

2) Declare três funções no módulo Main (exercício 1) para calcular a área de três figuras geométricas: quadrado (lado*lado), retângulo (base*altura) e triângulo ((base*altura)/2). Em seguida, chame essas funções no corpo do módulo Main. O programa deve apresentar a seguinte saída:

```
Área do quadrado: <resultado>
Área do retângulo: <resultado>
Área do triângulo: <resultado>
```

→ Lembre-se de usar a convenção recomendada para nomes de função e nomes de parâmetro.

3) Crie um módulo chamado de figuras geométricas e salve em um script. Em seguida, execute os seguintes passos:
   a. Adicione a declaração das três funções que você criou no Exercício 2;
   b. Adicione a assinatura de tipo para as três funções;
   c. Teste o seu script usando o ambiente interativo GHCi:
      i. Carregue o novo script:
         `:l <nome do script>`
      ii. Teste as três funções
         `<nome da funcao> <argumentos>`

4) Altere o módulo Main (exercício 2) seguindo os seguintes passos:
   a. Remova a declaração das três funções
   b. Adicione o ***import*** para o novo módulo (figuras geométricas)
   c. Execute o seu programa

→ Altere o nome do novo módulo de forma que fique diferente do nome do script e observe se seu programa continua funcionando. Por exemplo:
- nome do módulo: **figuras**
- nome do script: **figuras geométricas**

5) Vamos praticar agora função recursiva em Haskell. Para isso, crie um modulo chamado recursão e declare as seguintes funções recursivas:
   a. Somar todos os números reais no intervalo [0, y], sendo y > 0
      - Implemente a versão com if-then-else (soma_valores_IF)
      - Implemente a versão com guardas (soma_valores_G)
      - Implemente a versão sem usar if-the-else ou guardas (soma_valores)

   b. Somar todos os números reais no intervalo [x, y], sendo x < y
      - Implemente a versão com if-then-else (somatorio_IF)
      - Implemente a versão com guardas (somatorio_G)
      - **Desafio**: é possível implementar a versão sem usar if-then-else ou guardas (somatorio)?

   Em seguida, teste as funções que você criou:
   a. Usando o ambiente interativo GHCi
   b. No módulo Main

6) Vamos criar um módulo chamado "cálculo da media" dividido em três partes. Lembre-se de adicionar comentários a cada parte do seu código.
   a. Base de dados
      Nessa parte, declare 5 funções para representar os números reais de uma sequência. Por exemplo:

```
num 1 = 5.0
num 2 = 10.0
...
```

(Lê-se: a função *num* quando recebe *1* como entrada retorna *5.0*)
(Lê-se: a função *num* quando recebe *2* como entrada retorna *10.0*)

   b. Função recursiva (soma)
      Implemente uma função recursiva para somar todos os números retornados pelas funções da base de dados (num 1 + num 2 + ... num 5). A função, de preferência, não deve usar if-then-else ou guardas.

   c. Função (media)
      Implemente uma função (não recursiva) para calcular a média dos números retornados pelas funções da base de dados. Dica: a função ***fromIntegral*** converte um inteiro para real e possui a seguinte assinatura de tipo:

```
fromIntegral :: Int -> Float
```

   Em seguida, execute o programa que você criou:
   a. Usando o ambiente interativo GHCi. Por exemplo:

```
> :l CalculoMedia
> media 5
```

   b. No módulo Main
      - Faça o import do módulo *cálculo da média*
      - Chame a função *media <argumento>*