PROGRAMAÇÃO FUNCIONAL [BCC222]

Teste 4 (2018–1)
23 de maio de 2018
Matrícula: _____
Nome: _____

Departamento de Computação
Universidade Federal de Ouro Preto
Prof. José Romildo Malaquias

1 Jogo Adivinha o Número

Ao executar as tarefas que se segem você estará escrevendo uma 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 1 a 1000.
2. O programa exibe a mensagem `Adivinhe um número entre 1 e 1000`.
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`, e
   - permite que o usuário escolha se quer jogar novamente, e joga novamente em caso afirmativo.

Exemplo de execução da aplicação

```
$ ./advinha
Adivinha o número v1.0
======================================
Digite um número entre 1 e 1000: 444
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 200
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 111
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 157
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 138
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 123
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 130
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 125
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 128
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 129
Parabéns, você acertou

Deseja jogar novamente? n
```

Exercise 1

Em um arquivo `adivinha.hs` defina o módulo `Main` exportando a variável `main`.

Exercise 2

Defina `main` como uma ação de E/S que, quando executada:

- configura o sistema para não realizar **bufferização** da saída de dados padrão, e
- exibe uma mensagem identificando o programa e sua versão

Exemplo de execução da aplicação

```
*Main> main
Adivinha o número v1.0
======================================
```

Exercise 3

Defina uma função `simOuNao` que recebe uma string e resulta em uma ação de E/S que, quando executada:

- exibe a string na saída padrão (com o objetivo de fazer uma pergunta do tipo *sim ou não* ao usuário)
- lê a resposta do usuário
- verifica se a resposta é
  - *s* ou *S*, retornando verdadeiro
  - *n* ou *N*, retornando falso
  - qualquer outra coisa, chamando `simOuNao` novamente para que o usuário responda corretamente.

  Use uma expressão `case`.

Exemplo de execução da aplicação

```
*Main> simOuNao "Quer jogar novamente?"
Quer jogar novamente? talvez
Quer jogar novamente? k
Quer jogar novamente? S
True

*Main> simOuNao "Você é inteligente?"
Você é inteligente? com certeza
Você é inteligente?
Você é inteligente? acho que sim
Você é inteligente? n
False
```

Esta função deve ser usada em `jogar` (veja a tarefa 5) para verificar se o usuário deseja continuar jogando ou não.

Exercise 4

Defina uma função `acertar` que recebe um número a ser adivinhado e resulta em uma ação de E/S que, quando executada:

- exibe uma mensagem solicitando um número entre 1 e 1000
- lê o número informado pelo usuário
- compara o número informado com o número a ser adivinhado:
  - se forem iguais, exibe uma mensagem parabenizando o usuário por ter adivinhado o número
  - caso contrário
    - exibe uma mensagem informando que o número é muito pequeno ou muito grande, adequadamente
    - exibe uma mensagem solicitando ao usuário uma nova tentativa
    - faz uma nova tentativa através de uma chamada recursiva de `acertar`

Exemplo de execução da aplicação

```
*Main> acertar 119
Digite um número entre 1 e 1000: 600
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 23
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 119
Parabéns, você acertou
```

A função `acertar` deverá ser usada na definição de `jogar` (veja a tarefa 5).

Exercise 5

O programa deve permitir ao usuário jogar várias vezes, o que nos leva à necessidade do uso de recursão.

Defina uma ação de E/S `jogar` que, quando executada

- gera um número inteiro aleatório entre 1 e 1000, inclusive
- interage com o usuário até que o usuário acerte o número (veja a tarefa 4)
- verifica se o usuário deseja jogar novamente (veja a tarefa 3)
  - se sim, executa `jogar` recursivamente
  - se não, não faz nada

Para gerar um número aleatório, utilize a função `randomRIO` do módulo `System.Random`. A classe `Random` é formada pelos tipos para os quais pode-se gerar valores aleatórios. Os tipos inteiros `Int` e `Integer` são instâncias desta classe.

A função `randomRIO :: Random a => (a, a) -> IO a` recebe um par de valores como argumento e resulta em uma ação de E/S que, quando executada, gera e retorna um número pseudo-aleatório no intervalo fechado definido pelo par.

Exemplo de execução da aplicação

```
*Main> jogar
Digite um número entre 1 e 1000: 509
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 780
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 640
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 700
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 744
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 730
Muito grande
Tente novamente

Digite um número entre 1 e 1000: 720
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 725
Muito pequeno
Tente novamente

Digite um número entre 1 e 1000: 728
Parabéns, você acertou

Deseja jogar novamente?  n
```

A ação `jogar` deve ser usada em `main` para que o usuário possa joagar o jogo.

Exercise 6

Modifique o programa `adivinha.hs` de forma que quando o palpite do usuário estiver errado, o programa informe a faixa os limites inferior e superior do intervalo considerando os valores que o usuário já tenha digitado.

Sugestão: modifique a função `acertar` acrescentando dois argumentos correspondentes aos limites inferior e superior do intervalo

Exemplo de execução da aplicação

```
*Main> jogar
Digite um número entre 1 e 1000: 509
Muito pequeno
Tente novamente

Digite um número entre 510 e 1000: 780
Muito grande
Tente novamente

Digite um número entre 510 e 779: 640
Muito pequeno
Tente novamente

Digite um número entre 641 e 779: 700
Muito pequeno
Tente novamente

Digite um número entre 701 e 779: 744
Muito grande
Tente novamente

Digite um número entre 701 e 743: 730
Muito grande
Tente novamente

Digite um número entre 701 e 729: 720
Muito pequeno
Tente novamente

Digite um número entre 721 e 729: 725
Muito pequeno
Tente novamente

Digite um número entre 726 e 729: 728
Parabéns, você acertou

Deseja jogar novamente?  n
```

Exercise 7

Modifique o programa `adivinha.hs` de forma que o usuário possa especificar o intervalo a ser utilizado para adivinhar o número através de dois argumentos na linha de comando.

Exercise 8

Modifique o programa `adivinha.hs` para que seja exibido o número de tentativas feitas pelo usuário.

Exercise 9

Modifique o programa `adivinha.hs` para que o número máximo de tentativas feitas pelo usuário possa ser limitado. O número máximo de tentativas deverá ser informado na linha de comando. Caso ele não seja informado não haverá um limite máximo de tentativas.