INF 05008 - Fundamentos de Algoritmos 2021/1
Lista de exercícios 10 - Estruturas com Referência Mútua
Professor responsável: João Marcos Flach - jmflach@inf.ufrgs.br

Instruções

- Utilize o **modelo de arquivo** fornecido no *Moodle*. É obrigatório utilizar as definições de dados que estão no modelo. Mantenham a estrutura do modelo fornecido e façam as questões nos espaços delimitados para cada uma.
- Nas questões que novos tipos são criados, **deve ser colocada a definição do novo tipo criado**.
- A ordem dos argumentos das funções deve ser sempre a mesma ordem que os argumentos são elencados no enunciado da questão.

Exercícios

Nesta lista, vamos trabalhar com uma abstração de um sistema de arquivos utilizado por um computador, utilizando árvores como estrutura de dados. Como vocês devem saber, um sistema de arquivos é basicamente composto por **Diretórios** e **Arquivos**. Cada Diretório, além de um nome, contém um número arbitrário de Diretórios e de Arquivos, que chamaremos de **Conteúdo** do Diretório. Essas definições de dados já estão no modelo fornecido.
A seguir, temos uma imagem demonstrando a estrutura de arquivos de um diretório com nome "João":

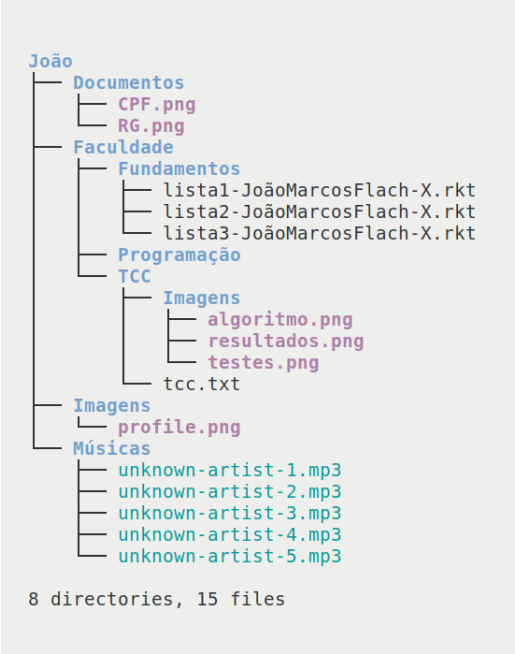

Figura 1: Diretório "João" - Os nomes em azul são diretórios e os outros são arquivos.

Exercício 1. Defina a constante `PASTA-JOÃO`, para representar a árvore da imagem anterior. Assuma que arquivos de imagem (.png) tem o tamanho 100 Kb, os arquivos de texto e do racket (.txt e .rkt) tem tamanho 50 Kb e as músicas (.mp3) tem tamanho 2 mb (2000 Kb). Utilizem os mesmos nomes e a mesma ordem de diretórios e arquivos para construir a árvore.
Como exemplo, a pasta `Documentos` seria:

```
(make-diretorio "Documentos" (list
                               (make-arquivo "CPF.png" 100)
                               (make-arquivo "RG.png" 100)))
```

Exercício 2. Construa a função `arquivo-no-dir?` que, dados o **Conteúdo de um diretório** e um **nome de arquivo**, verifica se existe um arquivo com este nome neste conteúdo, sem considerar subdiretórios. A função deve devolver o valor booleano correspondente.

Dica : Lembre-se que o Racket fornece automaticamente funções para verificação de tipos criados pelo usuário, como a função `arquivo?` que, dada uma variável, retorna `#true` se ela for do tipo Arquivo, e `#false` caso contrário.

Exercício 3. Construa a função `arquivo-encontrado?` que, dados o **Conteúdo de um diretório** e um **nome de arquivo**, verifica se existe um arquivo com este nome neste conteúdo, considerando subdiretórios, ou seja, a função retorna verdadeiro se um arquivo com o nome dado existe na lista de conteúdo recebida, ou em qualquer diretório da lista. A função deve devolver o valor booleano correspondente.

Exercício 4. Defina a função chamada `calcula-tamanho` que, dado um **Diretório**, calcula o tamanho necessário em disco para armazenar este diretório. O tamanho é calculado da seguinte forma: para cada arquivo que faz parte do diretório, soma-se o tamanho do arquivo, e para cada diretório soma-se 10 (para guardar as informações sobre o diretório) ao tamanho de seu conteúdo. O tamanho deve considerar toda a árvore do diretório (ou seja, o diretório e todos seus subdiretórios).

Exercício 5. Construa a função `mostra-caminho` que, dado um **nome de arquivo** e um **Diretório**, mostra o caminho para encontrar esse arquivo, ou uma mensagem dizendo que o arquivo não esta nesse diretório. O resultado deve ser uma string mostrando um caminho até o arquivo conforme os exemplos abaixo (podem existir mais de um caminho, podem escolher qualquer um para mostrar) ou a mensagem "Arquivo não encontrado". Exemplos:

```
(mostra-caminho "CPF.png" PASTA-JOÃO) -> "João -> Documentos -> CPF.png"
(mostra-caminho "CNPJ.png" PASTA-JOÃO) -> "Arquivo não encontrado"
(mostra-caminho "algoritmo.png" PASTA-JOÃO)
-> "João -> Faculdade -> TCC -> Imagens -> algoritmo.png"
```