Princípios de Programação

Trabalho Quinto

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2020/2021

Uma tarefa frequente que os engenheiros informáticos encontram pela frente é procurar dados relevantes em grandes ficheiros de texto. Uma forma procurar os dados pretendidos é começar com o ficheiro todo e filtrar progressivamente os dados.
Neste trabalho pretende-se desenvolver uma ferramenta de linha de comandos que nos ajude nesta tarefa.

A. StackGrep Vamos escrever um programa Haskell que receba um ficheiro de texto como único parâmetro na linha de comandos. O programa deverá apresentar todas as linhas desse ficheiro e pedir ao utilizador porque *string* pretende filtrar.

```
$ runhaskell StackGrep.hs teste.txt
[haskell] Data.List
[haskell] System.Environment
[java] java.util.List
[java] java.util.ArrayList
[java] java.io.File

Filtering:
> _
```

Neste momento, o programa deve aceitar qualquer *string* como input. Caso essa *string* seja diferente de `"pop"` (esta *string* tem um papel especial, explicado abaixo), iremos repetir o processo, mostrando apenas as linhas do ficheiro que contenham a *string*:

```
Filtering:
> java
```

```
[java] java.util.List
[java] java.util.ArrayList
[java] java.io.File

Filtering: java
> _
```

Podemos agora repetir o processo, filtrando desta vez por por `util`:

```
> util
[java] java.util.List
[java] java.util.ArrayList

Filtering: java, util
> _
```

Desta vez vamos inserir o comando especial `"pop"`. Ao fazê-lo o programa vai retirar o último filtro aplicado:

```
Filtering: java, util
> pop
[java] java.util.List
[java] java.util.ArrayList
[java] java.io.File

Filtering: java
> _
```

Agora podemos voltar a adicionar um filtro, ou anular (desempilhar) o último filtro. Optamos pela segunda.

```
Filtering: java
> pop
[haskell] Data.List
[haskell] System.Environment
[java] java.util.List
[java] java.util.ArrayList
[java] java.io.File

Filtering:
> _
```

Um último `"pop"` termina o programa.

Notas

1. Deverá submeter um ficheiro zip com o nome `t5_fcXXXXX.hs`, onde `XXXXX` é o seu número de aluno. Este ficheiro hs deverá conter a função main.

2. Os trabalhos serão avaliados automaticamente. Respeite os nomes do ficheiros e uso esperado.
3. Cada função (ou expressão) que escrever deverá vir sempre acompanhada de uma assinatura. Isto é válido para as funções enunciadas acima bem como para outras funções ajudantes que decidir implementar.
4. Para resolver este exercício pode utilizar toda a matéria do livro de texto até ao capítulo sobre Input-Output.
5. Devem usar e abusar das bibliotecas *standard* do Haskell e funções de ordem superior se acharem necessário.
6. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada.

Entrega. Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 21 de dezembro de 2020.

Plágio. Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Relembramos aqui um excerto da sinopse: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".