Universidade do Minho
Escola de Engenharia
Departamento de Informática

Projecto de Laboratórios de Informática I

Licenciatura em Engenharia Informática

1º Ano — 1º Semestre

Ano Lectivo 2021/2022

— Fase 2 de 2 —

Data de Lançamento: 28 de Novembro de 2021

Data Limite de Entrega: 08 de Janeiro de 2022

Novembro de 2021

1 Introdução

Neste enunciado apresentam-se as tarefas referentes à segunda (e última) fase do Projecto de Laboratórios de Informática I 2021/2022. O projecto será desenvolvido pelos **mesmos grupos** constituídos para a primeira fase e consiste novamente na resolução de um pequeno conjunto de *Tarefas*, a ser resolvido tendo por base o trabalho já elaborado pelo grupo durante a primeira fase.

2 Tarefas

Cada uma das seguintes tarefas deverá ser implementada num módulo (e, por conseguinte, ficheiro) próprio, à semelhança dos módulos usados para as tarefas da primeira fase.

2.1 Tarefa 5 – Aplicação gráfica completa

O objectivo desta tarefa consiste em aproveitar todas as funcionalidades já elaboradas nas outras *Tarefas* e construir uma aplicação gráfica que permita um utilizador jogar. Para o interface gráfico deverá utilizar a biblioteca *Gloss*[1], que terá oportunidade de estudar durante as aulas práticas.

O grafismo e as funcionalidades disponibilizadas ficam à criatividade dos alunos, sendo que, **no mínimo**, a aplicação deverá:

- Mostrar um menu inicial, nas quais deve incluir a opção para jogar um puzzle.
- Uma vez escolhida a opção de jogar, deverá carregar um puzzle e permitir ao jogador manipular (através do teclado) o personagem, detectando quando este atinge a porta. Ao atingir a porta deverá realizar alguma acção – por exemplo, avançar para outro puzzle, mostrar uma mensagem ou voltar para o menu inicial.

Sugestões de funcionalidades que serão valorizadas:

- Carregar um nível do sistema de ficheiros – por exemplo, através de um menu de navegação.
- Guardar um jogo em progresso (vulgo *save game*) e continuá-lo mais tarde (na mesma posição).

[1] `https://hackage.haskell.org/package/gloss`

- Manter uma pontuação (vulgo *highscore*) para cada nível jogado e jogador, sendo que quanto menor o número de movimentos para resolver um jogo, melhor será a pontuação.
- Assumindo implementada a *Tarefa 6*, permitir ao jogador escolher um puzzle e animar-lhe uma solução, por exemplo, executando, passo-a-passo e a cada segundo, cada movimento da sequência de movimentos que resolve o puzzle.
- Disponibilizar um editor de mapas.
- *Etc.*

Poderá alterar o código (*e.g.* tipos de dados) dado e/ou desenvolvido na primeira fase, desde que não quebre as funcionalidades implementadas nem os respectivos testes.

2.2 Tarefa 6 – Resolução de um puzzle

O objectivo desta tarefa é implementar a função

```
resolveJogo :: Int -> Jogo -> Maybe [Movimento]
```

que *tenta* (daí o `Maybe` no resultado) resolver um jogo num número máximo de movimentos. Resolver um jogo consiste em encontrar uma sequência de movimentos que o jogador pode realizar para chegar à porta. Sempre que não for possível encontrar uma sequência de movimentos vencedora dentro do limite de movimentos máximo imposto, a função deverá avaliar em `Nothing`.

Figura 1: Exemplo de uma situação de jogo.

A título de exemplo, considere o jogo (que denotaremos doravante pelo identificador `jogoEx`) representado na Figura 1. Como é fácil de perceber,

bastaria realizar a seguinte sequência de movimentos para vencer (resolver) o jogo: `Trepar`, `AndarEsquerda`, `AndarEsquerda`. Portanto, um output válido para a expressão `resolveJogo 3 jogoEx` seria:

```
Just [Trepar, AndarEsquerda, AndarEsquerda]
```

Caso o limite de movimentos fosse maior ainda (*e.g.* 5), outra sequência vencedora poderia também ser: `AndarDireita`, `AndarEsquerda`, `Trepar`, `AndarEsquerda`, `AndarEsquerda`. Por outro lado, caso o limite fosse menor ainda (*e.g.* 2), não existiria nenhuma sequência vencedora, uma vez que 3 é o número mínimo de movimentos necessários a realizar para levar o jogador à porta neste puzzle. Por corolário, quando o limite de movimentos imposto é 0 (zero), a função resume-se a testar se o jogo já se encontra resolvido (caso em que avaliaria em `Just []`) ou não (caso em que avaliaria em `Nothing`).

Nesta tarefa serão valorizadas estratégias eficientes que procurem a menor sequência de movimentos possível.

3 Entrega e Avaliação

A data limite para conclusão de todas as tarefas desta segunda fase é de **08 de Janeiro de 2022** e a respectiva avaliação terá um peso de **45%** na nota final da UC. A submissão será feita automaticamente através do GitLab onde, nessa data, será feita uma cópia do repositório de cada grupo, sendo apenas consideradas para avaliação os programas e demais artefactos que se encontrem no repositório nesse momento. O conteúdo dos repositórios será processado por ferramentas de detecção de plágio e, na eventualidade de serem detectadas cópias, estas serão consideradas **fraude** dando-se-lhes tratamento consequente.

Para além dos programas *Haskell* relativos às tarefas será considerada parte integrante do projecto todo o material de suporte à sua realização armazenado no repositório do respectivo grupo (código, documentação, ficheiros de teste, *etc.*). A utilização das diferentes ferramentas abordadas no curso (como *Haddock* ou *git*) deve seguir as recomendações enunciadas nas respectivas sessões laboratoriais. A avaliação desta fase do projecto terá em linha de conta todo esse material, atribuindo-lhe os seguintes pesos relativos:

Componente	**Peso**
Avaliação automática e qualitativa das Tarefas 5 e 6	80%
Documentação do código	10%
Quantidade e qualidade dos testes	5%
Utilização do sistema de versões	5%

A nota final será atribuída **independentemente** a cada membro do grupo em função da respectiva prestação. A avaliação automática será feita através de um conjunto de testes que não serão revelados aos grupos. A avaliação qualitativa incidirá sobre aspectos da implementação não passíveis de serem avaliados automaticamente (como a estrutura do código ou elegância da solução implementada).

A avaliação global do projecto incluirá uma discussão do trabalho realizado, numa sessão presencial, que poderá incluir (re)escrita de código.