Projecto de Laboratórios de Informática I
(1ª fase*)

Sokoban em *Haskell*

2015/2016 — LEI

1 Introdução

Neste enunciado apresentam-se as tarefas referentes à primeira fase do projecto da unidade curricular de Laboratórios de Informática I. O projecto será desenvolvido por grupos de 2 elementos, e consiste em pequenas aplicações *Haskell* que deverão responder a diferentes tarefas (apresentadas adiante).

O projecto baseia-se no puzzle *Sokoban* (`http://wikipedia.org/wiki/Sokoban`) onde se controla um *boneco* numa arrecadação por intermédio de comandos muito simples com o objectivo de empurrar as caixas para posições determinadas. Convidam-se os alunos a jogar a versão do jogo *online* (`http://sokoban.info`) para se familiarizarem com as regras do jogo.

No *BlackBoard* da disciplina, será mantida uma FAQ contendo respostas a questões e esclarecimentos que surjam ao longo do periodo de execução do projecto.

2 Entrada/Saída de Dados

Nas diferentes tarefas computacionais realizadas nesta fase do projecto, o formato para *entrada* e *saída* de dados é sempre textual. Os programas realizados irão ler os dados do `stdin`, e escrever os resultados em `stdout`.

2.1 Formato de Entrada

O formato de entrada compreende um preâmbulo comum a todas as tarefas com a informação relativa ao mapa da arrecadação e às posições do boneco

*Última actualização: 12 de Outubro de 2015

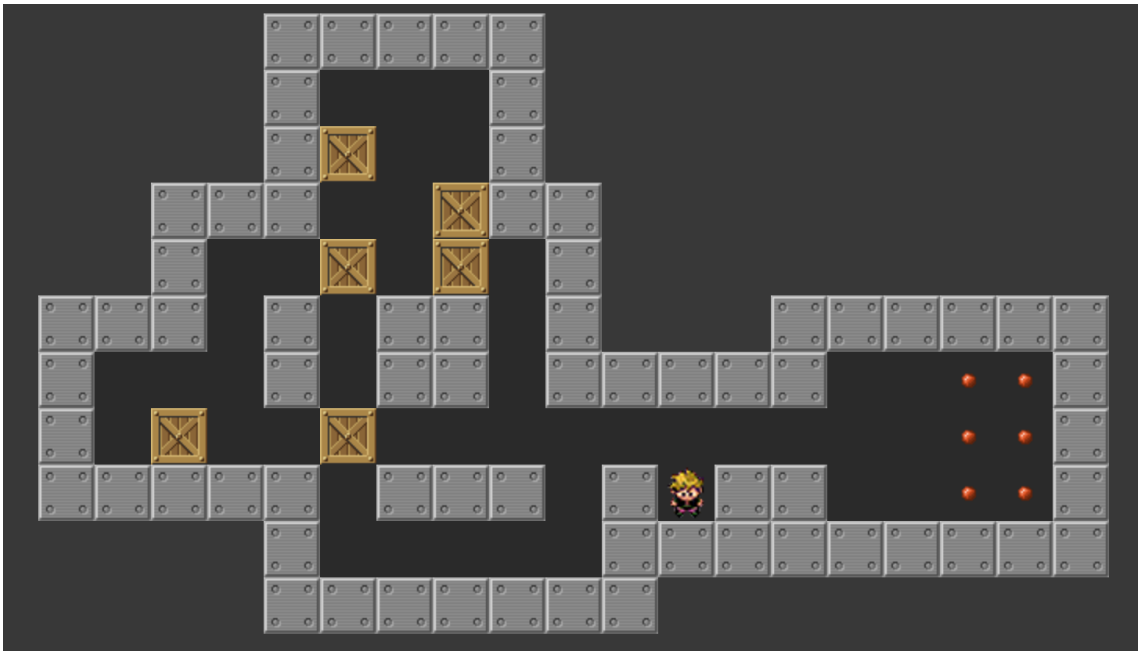

Figura 1: Planta do jogo *Sokoban*

e das caixas.

- A planta da arrecadação é sempre um rectangulo construido com os caracteres:
  - '`#`' – parede
  - ' ' (espaço) – área livre
  - '`.`' – local de arrumação
- Os contornos desse rectangulo devem sempre ser paredes (`#`).
- Após o mapa da arrecadação, segue-se uma linha com a posição inicial do boneco com o formato

  $$\langle \text{x\_pos} \rangle \ \langle \text{y\_pos} \rangle$$

  onde `x_pos` e `y_pos` são inteiros positivos (considera-se que o canto inferior esquerdo da planta corresponde à coordenada $(0, 0)$).
- Seguindo-se uma linha com as coordenadas de cada uma das caixas com um formato análogo ao da posição do boneco. Deverão existir tantas caixas quantos os locais de arrumação disponíveis.

A título de exemplo, os dados de entrada para a configuração apresentada na Figura 1 corresponderá o seguinte texto de entrada:

```
####################
#####   ###########
#####   ###########
#####   ###########
###      ##########
### # ## ##########
#   # ## #####  ..#
#                ..#
##### ### # ##  ..#
#####     #########
####################
11 2
5 8
7 7
5 6
7 6
2 3
5 3
```

Note que foram adicionados blocos '`#`' ao desenho apresentado na Figura 1 para se obter um rectangulo.

Dependendo da tarefa, poderá existir ainda uma linha adicional contendo comandos específicos para a tarefa. O formato dessa linha é descrito na apresentação da tarefa respectiva.

2.2 Formato de Saída

O formato de saída será específico para cada uma das tarefas propostas.

3 Tarefas

Nesta fase do projecto serão consideradas três tarefas computacionais. Estas tarefas correspondem aos problemas disponibilizados na plataforma `mooshak` (`http://mooshak.di.uminho.pt`), onde serão submetidas as respectivas soluções.

3.1 Validação do *Input*

Pretende-se nesta tarefa realizar um programa que permita validar se o *input* fornecido cumpre os requisitos impostos pela descrição apresentada na

secção anterior. O programa desenvolvido deve imprimir uma única linha de resultado contendo:

- `OK` — se o formato do *input* estiver de acordo com a descrição apresentada na Secção 2.1; ou
- `<num>` — quando for encontrado um erro. Nesse caso, `<num>` correponde ao número da linha onde for encontrada a primeira divergência com o formato prescrito.

O objectivo desta primeira tarefa é fundamentalmente o de ambientar os grupos com o sistema de submissão/avaliação da plataforma `mooshak`. Será por isso dedicada uma das aulas da UC à sua realização, possibilitando dessa forma aos grupos de trabalho interagirem com o sistema e interpretarem o *feedback* que lhes é dado.

3.2 Visualisação da *planta*

Nesta tarefa pretende-se produzir como resultado a visualização do mapa do jogo, i.e. o mapa da arrecadação com as caixas e o boneco nas posições respectivas. Para tal, e para além dos caracteres considerados na Secção 2.1, deve ainda considerar:

- '`o`' – boneco;
- '`I`' – caixa numa posição final;
- '`H`' – caixa numa posição não final.

Para tornar a representação da planta da arrecadação "menos pesada", iremos ainda remover os caracteres '`#`' que forem redundantes (i.e. quando todos à sua volta são também '`#`'). Assim, e a título de exemplo, a representação da configuração apresentada resultaria em:

```
    #####
    #   #
    #H  #
  ###  H##
  #  H H #
### # ## #   ######
#   # ## #####  ..#
# H  H          ..#
##### ### #o##  ..#
    #     #########
    #######
```

Na realização desta tarefa devemos assumir que os dados de entrada se encontram efectivamente de acordo com a especificação apresentada na Secção 2.1.

3.3 Cálculo do próximo estado

Pretende-se implementar um programa que determine qual a posição do *boneco* após a execução de um comando. Os comandos disponíveis correspondem às direcções possíveis para movimentar o boneco, nomeadamente:

L (left) – mover o boneco para a esquerda;

U (up) – mover o boneco para cima;

R (right) – mover o boneco para a direita

D (down) – mover o boneco para baixo;

Deve ainda atender aos seguintes aspectos:

- após a descrição da configuração inicial, surge uma linha adicional contendo um único caracter correspondente ao comando a realizar;
- assume-se que os dados de entrada estão correctos (i.e. de acordo com a especificação apresentada na Secção 2.1);
- se o comando não for aplicável (i.e. se de acordo com as regras do jogo, não for possível ao boneco movimentar-se para a direcção pedida), o estado deve ficar inalterado.
- como resultado devem ser impressas as coordenadas da nova posição seguinte do *boneco*. Para tal, deverá utilizar um formato análogo à da posição inicial, i.e. numa única linha contendo

$$\langle \text{x\_pos} \rangle \ \langle \text{y\_pos} \rangle$$

4 Entrega e Avaliação

A data limite para entrega de todas as componentes desta primeira fase do projecto é **15 de Novembro de 2015**, e a respectiva avaliação terá um peso de 40% na nota final da UC. As tarefas computacionais deverão ser submetidas na plataforma `mooshak`, sendo que estas serão desde logo objecto de uma avaliação automática por parte da plataforma (com um peso

descriminado abaixo). Cada grupo é responsável por submeter na plataforma `mooshak` unicamente programas da sua autoria[1].

Para além dos programas submetidos na plataforma `mooshak`, será considerada parte integrante do projecto todo o material de suporte à sua realização armazenado no repositório SVN do respectivo grupo (código, documentação, ficheiros de teste, etc.). A utilização das diferentes ferramentas abordadas no curso (como `haddock`; *SVN*; etc.) 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:

Avaliação automática das tarefas computacionais $(15 + 10 + 15)$	40%
Avaliação qualitativa das tarefas computacionais e do processo de desenvolvimento	20%
Utilização do SVN e estrutura do repositório	10%
Quantidade e qualidade dos testes	20%
Documentação do código	10%

A nota final é atribuída independentemente a cada membro do grupo em função da respectiva prestação.

[1]Os programas submetidos irão ser processados por ferramentas de detecção de plágio e, na eventualidade serem detectadas cópias, estas serão consideradas fraude dando-se-lhes tratamento consequente.