Trabalho da Disciplina Laboratório de Programação I
FEN06 04049
Prof. João Araujo

A Pequena donzela encarcerada

Existe uma antiga lenda japonesa sobre a "*hakoiri musume*", ou seja, a pequena donzela, que não sabe nada sobre o mundo, que está encarcerada numa caixa de ilusões. Seu caminho para a liberdade é impedido por sua mãe, seu pai, irmãos e irmãs.... Enquanto ela não sair desta caixa, não conhecerá a luz da verdade.

Seu trabalho de programação deste semestre é ajudar a pobre *hakoiri musume* a encontrar seu caminho até a luz! Porém, neste mundo cheio de simbolismos da cultura japonesa, cada personagem será representado no computador por uma figura geométrica que corresponde à inocência do personagem, mais que a forma física. Assim a nossa pobre *hakoiri musume* é representada pela maior figura do tabuleiro, representando que sua inocência a impede de encontrar o caminho até a saída, enquanto sua família impõe dificuldades neste caminho.

As regras são simples: Você deve levar a donzela até a saída da caixa, deslocando as outras figuras para dar passagem. Neste nosso tabuleiro matemático, cada peça é representada por um número e as casas vazias são representadas por zero. O tabuleiro tem NxM casas com as posições iniciais das peças. Uma figura só pode ser movimentada se existirem casas livres suficientes para a movimentação e só pode ser movimentada na horizontal ou na vertical, nunca na diagonal. Seu objetivo é levar a donzela até a porta de saída.

Por exemplo, a configuração 1 do tipo:

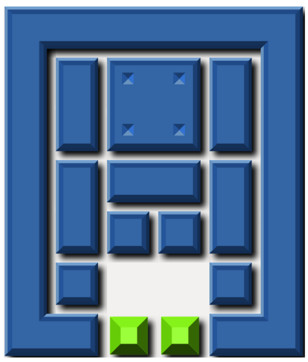

Figura 1: Configuração 1

Se atribuímos uma letra para cada tipo de peça, poderíamos ter:

```
*  *  *  *  *  *
*  a  D  D  b  *
*  a  D  D  b  *
*  c  d  d  e  *
*  c  g  h  e  *
*  f        i  *
*  *        *  *
```

Uma solução possível para este problema é o da força bruta, procurando todas as possibilidades até colocar a donzela na posição adequada do tabuleiro. Esta busca pode ser computacionalmente muito intensiva e você deve eliminar os caminhos já percorridos. Uma forma de diminuir este trabalho seria armazenar cada movimento possível numa árvore (não binária). Se cada possível configuração do tabuleiro possuir uma identificação única, se você atingir uma configuração que já se encontra na árvore, você pode continuar a partir deste ponto.

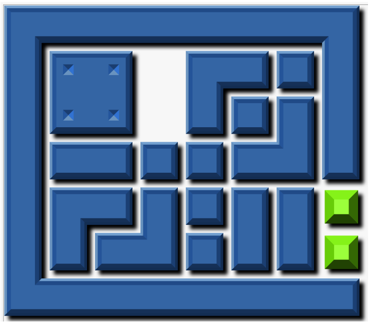

Figura 2: Configuração 2

A figura 2 possui a seguinte representação:

*	*	*	*	*	*	*	*
*	D	D		a	a	b	*
*	D	D		a	c	d	*
*	e	e	f	g	d	d	*
*	h	h	i	j	k	l	
*	h	i	i	m	k	l	
*	*	*	*	*	*	*	*

1 Fase 1:

Escreva um programa em C que receba argumentos da linha de comando que escolham a configuração 1 ou 2 e também direcionem uma das peças para um movimento. A saída de seu programa deve ser uma matriz com a peça pedida movimentada corretamente, se possível. Se a configuração for inválida, ou seja, diferente de 1 ou 2, deve ser emitida mensagem de erro. Também deve ser emitida mensagem de erro se não for possível movimentar a peça.

Os parâmetros são:

-cn: no qual **n** é a configuração 1 ou 2, ex. *-n1* ou *-n2*. Se for o único comando, deve imprimir a configuração pedida. O valor *default* , isto é, se não for especificada nenhuma configuração, é 1.

-m x y d: movimenta a peça que está nas coordenadas *(x,y)* para a direção *d*. As coordenadas x e y começam em 1 e não levam em conta as paredes do jogo. Assim, na configuração **1** as coordenadas vão de *1,1* até *5,4* e na configuração **2**, de *1,1* até *5,6*. O valor de **d** pode ser **T, B, E ou D**, que representam, respectivamente movimentos para o **T**opo, para **B**aixo, para **D**ireita ou **E**squerda. Após o comando, deve ser impressa a configuração inicial e a configuração após o movimento. Se não for possível movimentar a peça, deve ser emitida mensagem de erro.

1.1 Exemplos:

---

haikori
ou
haikori -c1
imprime na tela:

*	*	*	*	*	*
*	a	D	D	b	*
*	a	D	D	b	*
*	c	d	d	e	*
*	c	g	h	e	*
*	f			j	*
*	*			*	*

---

haikori -c2
imprime na tela:

*	*	*	*	*	*	*	*
*	D	D		a	a	b	*
*	D	D		a	c	d	*
*	e	e	f	g	d	d	*
*	h	h	i	j	k	l	
*	h	i	i	m	k	l	
*	*	*	*	*	*	*	*

---

haikori -c2 -m 1 1 D
ou
haikori -c2 -m 1 2 D
ou
haikori -c2 -m 2 1 D
ou
haikori -c2 -m 2 2 D

*	*	*	*	*	*	*	*
*	D	D		a	a	b	*
*	D	D		a	c	d	*
*	e	e	f	g	d	d	*
*	h	h	i	j	k	l	
*	h	i	i	m	k	l	
*	*	*	*	*	*	*	*

*	*	*	*	*	*	*	*
*		D	D	a	a	b	*
*		D	D	a	c	d	*
*	e	e	f	g	d	d	*
*	h	h	i	j	k	l	
*	h	i	i	m	k	l	
*	*	*	*	*	*	*	*

---

haikori -c2 -m 1 1 B
imprime:

*	*	*	*	*	*	*	*
*	D	D		a	a	b	*
*	D	D		a	c	d	*
*	e	e	f	g	d	d	*
*	h	h	i	j	k	l	
*	h	i	i	m	k	l	
*	*	*	*	*	*	*	*

Impossível movimentar peça em 1,1 para baixo

---

2 Fase 2:

mais detalhes na próxima semana

3 Fase 3:

mais detalhes na próxima semana

4 Fase 4:

mais detalhes na próxima semana