Universidade do Minho	**Escola de Engenharia da Universidade do Minho** **Mestrado Integrado em Eng. Electrónica Industrial e Computadores** **Programação de Computadores**	**2013/2014** **MIEEIC** **(1º Ano)** **1º Sem**
DOCENTE: Luís Paulo Reis		
FICHA DE EXERCÍCIOS 6: JOGO COM MATRIZES		

Algoritmos com Matrizes – Jogo Simples

```
0 0 0 0 0 0 0 0
0 1 0 0 2 1 0 0
0 0 0 2 2 2 0 0
0 0 0 0 0 0 0 1
1 1 2 0 0 0 0 1
1 0 0 0 0 0 0 0
0 0 0 1 0 0 1 0
0 0 0 0 1 0 0 0
```

6.1) Escreva uma função ***void preenche_tabuleiro(int tab[N][N], int val, int rep)*** que introduza numa matriz de dimensão N x N elementos (N definido com um define, ex: #define N 8) rep números inteiros com o valor Val, em posições aleatórias inicialmente vazias (incialmente com o valor 0).

6.2) Escreva um algoritmo que mostre no écran, devidamente formatada a matriz com N x N elementos ***void mostra_matriz(int tab[N][N]).***

6.3) Construa o programa principal que lhe permita testar as duas funções anteriores (criando uma matriz com 8x8 zeros e mostrando o seu conteúdo no écran). Em seguida deverá introduzir na matriz 10 elementos com o valor 1 e 5 elementos com o valor 2, em posições aleatórias e mostrar o conteúdo da matriz final. Vá actualizando o programa principal com testes adequados às restantes alíneas do exercício. O resultado final deverá ser semelhante ao apresentado na primeira matriz.

```
0 0 0 0 0 0 0 0
0 0 0 0 2 1 0 0
0 0 0 2 2 2 0 0
0 0 0 0 0 0 0 1
1 1 2 0 0 0 0 1
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

6.4) Escreva uma função que retorne o número de elementos da matriz com o valor x, ***int conta_elementos(int tab[N][N], int val)***. Teste a função contando o número de 1s e número de 2s na matriz, o que deverá retornar respetivamente 10 e 5 para o exemplo apresentado.

6.5) Escreva uma função ***void elimina_isolados(int tab[N][N], int val)*** que elimine da matriz, todos os elementos com valor val, que não tenham nenhum elemento adjacente (na horizontal ou vertical) com um valor diferente de 0, isto é, que estejam rodeados de zeros. Teste a função, eliminando todos os 1s isolados e visualizando o conteúdo final da matriz, semelhante ao apresentado na segunda matriz.

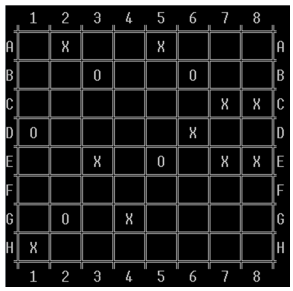
```
0 0 0 0 0 0 0 0
0 0 0 0 0 2 0 0
0 0 0 0 2 2 2 0
0 0 0 0 0 0 0 1
1 1 0 2 0 0 0 1
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

6.6) Escreva uma função ***void move_pecas1(int tab[N][N], int dirx, int diry, int val)*** que movimente todas as peças com valor val (1 ou 2) na direção dirx, diry (dirx e diry com valores, -1, 0 ou 1). Por exemplo se dirx = 1 e diry = 0 e val = 2, todas as peças do jogador 2 serão movimentadas para a direita. Caso uma peça se movimente para cima de outra peça, essa peça desaparece e caso se movimente para fora do tabuleiro a peça deseparece também.

```
0 0 0 0 0 0 0 0
0 0 0 0 0 2 0 0
0 0 0 0 2 2 2 0
0 0 0 0 0 0 0 1
1 0 1 2 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
```

6.7) Escreva uma função ***void move_pecas2(int tab[N][N], int dirx, int diry, int val)*** que movimente todas as peças com valor val (1 ou 2) na direção dirx, diry (dirx e diry com valores, -1, 0 ou 1). Por exemplo se dirx = 1 e diry = 0 e val = 1, todas as peças do jogador 1 serão movimentadas para a direita. Esta função difere da anterior pois as peças deverão ser movimentadas exclusivamente se a nova posição estiver livre e dentro do tabuleiro.

6.8) Crie uma função ***void mostra_tabuleiro(int tab[N][N])***que permita visualizar a matriz como se fosse um tabuleiro com peças (X e O) e com o formato indicado a seguir (ou melhor…). A função deve permitir visualizar tabuleiros com qualquer dimensão.



6.9) Teste as duas funções anteriores com exemplos significativos.

6.10) Crie um motor de jogo simples que funcione do seguinte modo. Em cada uma de 10 jogadas, adiciona à matriz (de 8x8) 10-NumJog peças de cada jogador, ou seja 9 peças na jogada 1, 8 na jogada 2, ... e 0 peças na última jogada. Em seguida elimina todas as peças isoladas de ambos os jogadores. Em seguida movimenta as peças do jogador 1 numa direção aleatória (horizontal ou vertical) uma casa, fazendo o mesmo logo em seguida para o jogador 2. Utilize inicialmente move_pecas1 e depois experimente move_pecas2 para esta tarefa. Em seguida elimina novamente os elementos isolados de ambos os jogadores. Nesta fase, caso um dos jogadores não tenha peças, o jogo termina com a vitória do outro jogador, ou empate (caso ambos não tenham peças). No final das 10 jogadas vence o jogador com mais peças (ou empate se tiverem o mesmo número de peças).

6.11) Experimente o jogo 1000 vezes (utilizando cada uma das funções move_pecas) e conclua sobre as probabilidades de cada um dos jogadores (1 e 2) vencerem em cada um dos casos…