----------------------------------------Identificação do trabalho (jogo) e do grupo------------------------------------------

Jogo: Moxie

Grupo: 3

Trabalho realizado por:

José Pedro Teixeira Ramos – up202005460

José Leandro Rodrigues da Silva – up202008051

Contribuição: 50/50

------------------------------------------------------Instalação e execução-------------------------------------------------------

Para executar o jogo:

consult(‘…/Game.pl’)

play.

--------------------------------------------------------Descrição do jogo-----------------------------------------------------------

Cada jogador começa com oito peças na reserva fora do tabuleiro. Uma jogada consiste em: 1) colocar uma peça da reserva em qualquer casa vazia; 2) mover uma peça que esteja em campo para qualquer quadrado adjacente, ortogonal ou diagonalmente (como um rei de xadrez), ou 3) capturar uma peça inimiga saltando sobre ela, como uma dama de 8 direções. Vários saltos são permitidos. Quando um salto é possível, o salto é obrigatório, embora o jogador possa escolher qualquer movimento de salto disponível.

O objetivo é formar uma linha de três com suas peças, ortogonal ou diagonalmente sem espaços intermediários, ou capturar seis peças inimigas (ou seja, reduzir as forças do inimigo a duas para que ele não possa vencer).

Fonte: https://boardgamegeek.com/boardgame/24546/moxie

--------------------------------------------------------Descrição do jogo-----------------------------------------------------------

Representação interna do estado do jogo: A variável GameState tem tudo que precisamos para fazer a representação interna do jogo, ela está estruturada da seguinte maneira [Arena, Peça\_atual, Peças\_restantes\_X, Peças\_restantes\_O]. A arena é uma lista de tamanho 16 que começa com espaços vazios ‘ ‘, e à medida que o jogo vai percorrendo vão se substituindo os espaços por peças, por ex ‘x’ ou ‘o’, de acordo com o input do jogador. A Peça\_atual é a peça que está a jogar atualmente, por ex ‘x’ ou ‘o’, começa sempre a jogar a peça ‘x’. As Peças\_restantes\_X e as Peças\_restantes\_O são o número de peças que restam em reserva e ainda podem ser colocadas em campo, sendo que Peças\_restantes\_X é o jogador X e o Peças\_restantes\_O é o jogador O, no início começam os dois com 8. Exemplo de representação do jogo inicial: Arena == [‘ ‘,‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘], Peça\_atual == ‘x’, Peças\_restantes\_X == 8 e Peças\_restantes\_O == 8.

Exemplo de representação do jogo intermédio: Arena == [‘ ‘,‘x‘, ‘x‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘o‘, ‘o‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘o‘, ‘ ‘], Peça\_atual == ‘o’, Peças\_restantes\_X == 2 e Peças\_restantes\_O == 5.

Exemplo de representação do jogo final: Arena == [‘ ‘,‘x‘, ‘x‘, ‘ ‘, ‘o‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘o‘, ‘o‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘ ‘, ‘o‘, ‘ ‘], Peça\_atual == ‘o’, Peças\_restantes\_X == 2 e Peças\_restantes\_O == 4.

Visualização do estado de jogo: Quando fazemos play. O prolog mostra-nos um menu com 1 - play, 2 - rules e 3 - exit, para interagir com o menu escrevemos o número daquilo que queremos, por exemplo 1. vai para o play. O play mostra os diferentes modos de jogo, o rules mostra as regras e o exit sai do jogo. No play ele faz o mesmo procedimento, ou seja, mostra vários modos de jogo e espera por input do jogador. Quando estamos num jogo temos a função display\_game(+GameState), no qual pega na Arena, Peças\_restantes\_X e Peças\_restantes\_O do GameState. No início ele chama a função remaining(Peça,+Peças\_restantes) para as duas peças, que repete a Peça pelo nr de Peças\_restantes, isto serve para representar o nrº de peças restantes de cada jogador. Depois disto ele chama a função arena\_x(+Arena, +Row), esta função pega na arena e transforma a lista Arena numa representação gráfica, pegando nos elementos um a um recursivamente até não ter mais elementos de maneira a que fica na seguinte maneira ‘x--', quando a Row é igual a 0 ele escreve ‘x’ ,nl, ‘| | | |’,nl e depois repõe a Row para os seu valor original.