

# Campo Bello

## Relatório Intercalar



Mestrado Integrado em Engenharia Informática e  
Computação

Programação em Lógica

**Grupo Campo Bello 1:**  
Francisca Cerquinho - up201505791  
Mariana Silva - up201506197

Faculdade de Engenharia da Universidade do Porto  
Rua Roberto Frias, 4200-465 Porto, Portugal

13 de Novembro de 2017

## Resumo

Este projeto consiste na implementação de um jogo de tabuleiro, o *Campo Bello*, através da linguagem de programação em lógica, Prolog, que é baseada num paradigma declarativo assente em três conceitos fundamentais: a unificação, a recursividade e o *backtracking*.

O principal objetivo é que cada jogador remova o maior número possível das suas próprias peças, até que não possua mais movimentos válidos ou mais peças.

O problema proposto foi na sua totalidade implementado, através da utilização dos predicados disponibilizados pelo *SICStus Prolog*, onde foi tido em conta não só a funcionalidade do próprio jogo, como também a eficiência do próprio código. Assim, foi possível a consolidação dos conceitos lecionados ao longo da unidade curricular.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>O Jogo Campo Bello</b>	<b>5</b>
2.1	História . . . . .	5
2.2	Objetivo e regras do jogo . . . . .	5
<b>3</b>	<b>Lógica do jogo</b>	<b>7</b>
3.1	Representação do Estado do Jogo . . . . .	7
3.1.1	Representação do estado inicial do tabuleiro . . . . .	7
3.2	Visualização do Tabuleiro . . . . .	7
3.3	Lista de Jogadas Válidas . . . . .	9
3.4	Execução de Jogadas . . . . .	9
3.5	Avaliação do Tabuleiro . . . . .	9
3.6	Final do Jogo . . . . .	9
3.7	Jogada do Computador . . . . .	10
<b>4</b>	<b>Interface com o Utilizador</b>	<b>11</b>
<b>5</b>	<b>Conclusões</b>	<b>13</b>
<b>6</b>	<b>Bibliografia</b>	<b>14</b>
<b>7</b>	<b>Anexos</b>	<b>15</b>

# 1 Introdução

No âmbito da unidade curricular Programação em Lógica foi-nos proposto a realização de um jogo de tabuleiro com base na linguagem de programação *Prolog*, pondo à prova os nossos conhecimentos relativamente a regras e problemas intrínsecos à linguagem.

Efetivamente, ao longo deste relatório irão ser abordados três grandes tópicos, nomeadamente, a história e regras do próprio jogo, a descrição dos principais predicados utilizados para a sua lógica, manipulação e visualização do tabuleiro e, por fim, a interface com o utilizador. Teremos em conta, também, no final, as principais conclusões deste projeto e possíveis melhorias.

Desta forma, procuramos responder ao que nos é exigido, de forma sucinta e explícita, sendo nossa intenção fazer com que o presente relatório sirva de guia e suporte para os interessados no jogo.

## 2 O Jogo Campo Bello

### 2.1 História

O *Campo Bello* foi pensado no início de 2014 por John Caddell e baseado no jogo *Cracker Barrel Peg*, adaptando-o para múltiplos jogadores. Depois de vários protótipos e dezenas de horas de teste, este jogo demorou mais de dois anos a ser implementado.

### 2.2 Objetivo e regras do jogo

Neste jogo, cada jogador tenta remover o maior número possível das suas peças do tabuleiro. Na sua vez, o jogador deve saltar com a sua peça para outra e se a peça que saltou for uma das suas, deve retirá-la do jogo. Caso contrário, se for uma das adversárias, então pode remover qualquer uma das suas peças do tabuleiro (incluindo a usada no salto).

Cada jogador pode "encadear" até 3 saltos com a mesma peça durante a sua vez, mas não pode saltar sobre a mesma duas vezes. A peça com que o jogador salta não pode ocupar o mesmo espaço durante a mesma jogada. Caso não consiga dar um salto, o jogador pode ignorar a sua vez. O jogo continua até que um jogador não tenha mais peças no tabuleiro ou nenhum jogador consiga fazer um salto válido.

No final do jogo, cada jogador obtém 1 ponto para cada uma das suas peças fora da sua área de partida e 3 pontos para cada peça que está na sua área de partida.

O jogador com o menor número de pontos ganha.

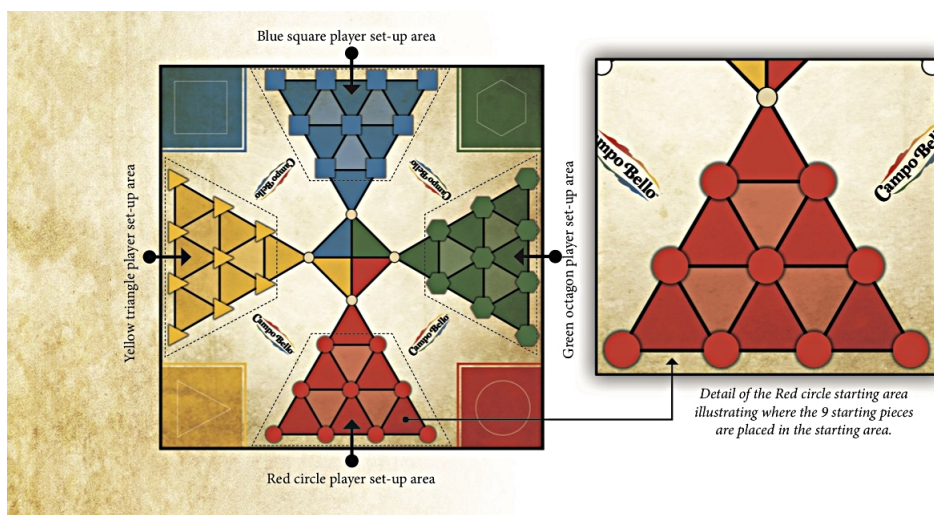


Figura 1: Aspeto do tabuleiro

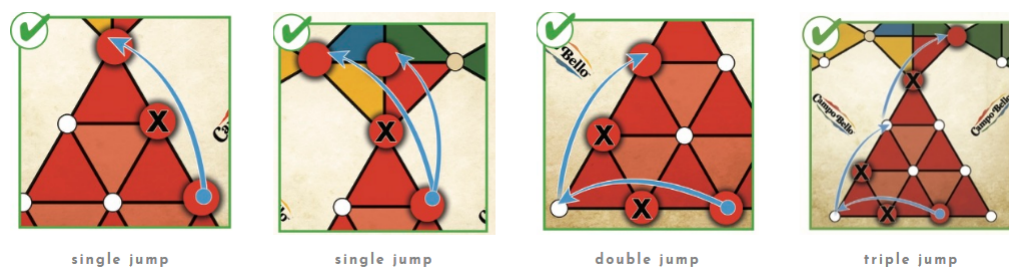


Figura 2: Movimentos permitidos



Figura 3: Movimentos não permitidos

## 3 Lógica do jogo

### 3.1 Representação do Estado do Jogo

A forma de representação do estado do tabuleiro usada foi uma lista com 9 listas. Nela estão presentes quatro tipos de peças diferentes, *pieceX1*, *pieceX2*, *pieceY1*, *pieceY2*, em que o X e Y distingue as peças de cada jogador e *noPiece* é um lugar do tabuleiro onde é possível mover as peças, mas onde naquele momento não está nenhuma.

#### 3.1.1 Representação do estado inicial do tabuleiro

```
1 initialBoard([[empty, pieceX1, pieceX1, pieceX1, pieceX1, empty,
2               empty, empty, empty],
3 [empty, empty, pieceX1, pieceX1, pieceX1, empty, empty, empty,
4   pieceY1],
5 [empty, empty, empty, pieceX1, pieceX1, empty, empty, pieceY1,
6   pieceY1],
7 [empty, empty, empty, empty, noPiece, empty, pieceY1, pieceY1,
8   pieceY1],
9 [pieceX2, pieceX2, pieceX2, noPiece, noPiece, noPiece, pieceY1,
10  pieceY1, pieceY1],
11 [pieceX2, pieceX2, pieceX2, empty, noPiece, empty, empty, empty,
12  empty],
13 [pieceX2, pieceX2, empty, empty, pieceY2, pieceY2, empty, empty,
14  empty],
15 [pieceX2, empty, empty, empty, pieceY2, pieceY2, pieceY2, empty,
16  empty],
17 [empty, empty, empty, empty, pieceY2, pieceY2, pieceY2, pieceY2,
18  empty]]).
```

### 3.2 Visualização do Tabuleiro

De forma a visualizarmos o tabuleiro, representado por uma matriz de nove linhas por nove colunas, foram construídos os seguintes predicados:

```
1 printFinalBoard([L|Ls]):-
2     nl,
3     printLetters, nl,
4     printBoard([L|Ls], 0),
5     printLine.
6
7 printLetters:-write('      A      B      C      D      E      F
8                 G      H      I').
9
10 printSpaces:-write('      |      |      |      |      |      |
11                  |      |      |').
12
13 printBoard([], _).
14 printBoard([L|Ls], Y) :-
15     printLine, nl,
16     printSpaces, nl,
17     Y1 is Y+1,
```

```

16         printFinalRow(L,Y1),nl,
17         printSpaces,nl,
18         printBoard(Ls,Y1).
19
20 printFinalRow([X|Xs],Y):-
21     write(Y),
22     write(' | '), printRow([X|Xs]).
23 printRow([X|Xs]):-
24     getSymbol(X,Piece),
25     write(' '), write(Piece),write(' | '),
26     printRow(Xs).
27 printRow([]).
28 printLine:-write('
-----').

```

A notar que cada tipo de peça é visualizada com um símbolo específico:

- *empty* representado com o símbolo ' ';
- *pieceX1* representado com o símbolo 'X';
- *pieceX2* representado com o símbolo 'X';
- *pieceY1* representado com o símbolo 'Y';
- *pieceY2* representado com o símbolo 'Y';
- *noPiece* representado com o símbolo 'N'.

```
| ?- initialBoard(B),printFinalBoard(B).
```

	A	B	C	D	E	F	G	H	I
1			X		X		X		
2				X		X		X	
3					X		X		Y
4						N		Y	
5		X		X		X		N	
6		X		X		X		N	
7		X		X				Y	
8		X						Y	
9								Y	

**Figura 4:** Imagem do tabuleiro correspondente ao output produzido pelo predicado de visualização



### 3.3 Lista de Jogadas Válidas

Em cada jogada, o utilizador deve inserir qual a peça que quer mover e quais as coordenadas de destino. De forma a validar a posição de origem, foi criado um predicado *validateSourcePiece*(-Ncol, -Nrow, -Board, -Piece) que verifica se o jogador escolheu uma das suas peças. A posição de destino da peça é testada no sentido de garantir que se trata de um movimento válido, através do predicado *validateDestinyPiece*(-LastCol, -LastRow, -Ncol, -Nrow, -Board, -Piece, -Area, -BoardOut) que invoca os predicados *checkIfCanMoveX*(-Ncol, -Nrow, -LastCol, -LastRow, -Board, -Piece, -BoardOut, -Area) e *checkIfCanMoveY*(-Ncol, -Nrow, -LastCol, -LastRow, -Board, -Piece, -BoardOut, -Area)) para validar os movimentos do jogador X e Y, respetivamente.

### 3.4 Execução de Jogadas

A partir do momento em que o jogador decide a sua jogada, através dos predicados *chooseSourceCoords*(+RowSource, +ColSource, -Board, -Piece) e *chooseDestinyCoords*(-RowSource, -ColSource, -Board, -Piece, +Area, +BoardOut) é verificado se o movimento é válido. Caso não seja validado, é pedida a inserção de novas posições, caso contrário é verificado se o jogador foi para uma posição onde não existiam peças. Se isso acontecer, o jogador pode saltar para uma nova posição e se esta estiver novamente vazia, o utilizador pode, pela última vez, saltar novamente, isto é, pode fazer saltos duplos ou triplos, pelo predicado *chooseNewJump*(-Board, +BoardOut, -LastColPiece, -LastRowPiece, -LastRow, -LastCol, +Row, +Col, -Piece, -Area, +Continue). Caso o jogador salte para uma posição onde exista uma peça sua, esta é removida do jogo se for uma peça do adversário, é perguntado ao utilizador qual a peça do tabuleiro que quer remover, através do predicado *choosePieceToRemove*(-Board, +BoardOut).

### 3.5 Avaliação do Tabuleiro

A avaliação e manipulação do tabuleiro foi conseguida através dos predicados *setPiece*(-BoardIn, -Nrow, -Ncol, -Piece, +BoardOut), *setOnRow*(Pos, [Row—Remainder], Ncol, Piece, [Row—Newremainder]), *setOnCol*(1, [—Remainder], Piece, [Piece—Remainder]) e *getElement*(-Board, -Nrow, -Ncol, +Element).

Para o nível de dificuldade elevada foi peremptória a avaliação do tabuleiro para que o computador comparasse todos os seus movimentos possíveis e escolhesse o melhor, isto é, aquele onde obtém o menor número de pontos. Esta avaliação é feita através dos predicados *evaluateBoards*(-Board, +Points) e *listOfBestMovements*(+FinalList, -Board) que avaliam todos os tabuleiros com todas as jogadas válidas e retornam uma lista ordenada, de forma crescente, pelo número de pontos seguida da posição. Assim, o primeiro elemento dessa lista é a melhor jogada que o computador pode efetuar. No entanto, não foi possível a sua completa implementação.

### 3.6 Final do Jogo

O jogo termina em duas situações diferentes, nomeadamente quando o jogador não possui mais movimentos válidos, pelo predicado *checkMoves*(-Piece,

*-Board*) ou quando não possui mais peças, através do predicado *checkPieces(-Piece, -Board)*. O predicado responsável por essa verificação dessas duas condições é o *endGame(-Board)*.

Posteriormente e de modo a identificar o vencedor foram criados os predicados *calculatePoints(-Board, +PointsX, +PointsY)* e *checkWinner(-Board, -PointsX, -PointsY)* que calculam os pontos de cada jogador e verificam qual o vencedor, respetivamente. O cálculo da pontuação é baseado no número de peças que cada jogador tem dentro e fora da sua área de partida, em que é obtido 1 ponto para cada peça fora e 3 pontos para cada peça dentro da sua área de partida. O vencedor é o jogador com o menor número de pontos.

### 3.7 Jogada do Computador

Se o utilizador escolher os modos Computador contra Jogador ou Computador contra Computador, ele poderá escolher qual o nível de dificuldade do jogo.

No nível de dificuldade normal, foram elaborados predicados responsáveis por retornar posições e movimentos válidos de forma aleatória. Para a escolha da peça que quer mover foi criado o predicado *listOfValidSourceMoveX(+Board, -FinalListX)* que cria uma lista com as peças do computador e que é percorrida no predicado *listOfPiecesThatHasPossibleMoveX(-FinalList, +Board)* que cria uma outra com apenas peças que têm movimentos possíveis. Posteriormente uma das peças é escolhida aleatoriamente através do predicado *random(+L, +U, -R)*. Relativamente à escolha do movimento, é criado o predicado *listOfValidDestinyMove(-List, +LastRow, +LastCol, +Area, +Board)* que cria uma lista com todos os destinos válidos da peça que o computador escolheu anteriormente e retorna, aleatoriamente, um desses destinos.

Pelo contrário no nível de dificuldade elevada é feita uma avaliação do tabuleiro por parte do computador para escolher qual a sua melhor jogada. Assim, o computador em vez de selecionar aleatoriamente uma peça válida, escolhe qual a que lhe permite obter um menor número de pontos. Os predicados utilizados foram os enunciados no tópico **Avaliação do Tabuleiro**. A notar que esta funcionalidade não foi implementada na totalidade com sucesso, pela tentativa infrutuíta de percorrer uma lista acedendo a elementos de outra.

## 4 Interface com o Utilizador

O módulo de interface com o utilizador é iniciado com um menu principal que permite ao utilizador nas opções 1., 2. e 3. jogar jogador contra jogador, *pc* contra jogador e *pc* contra *pc*, respetivamente. Na opção 4. aceder a um outro menu com as instruções de jogo e finalmente na opção 5., sair do jogo.

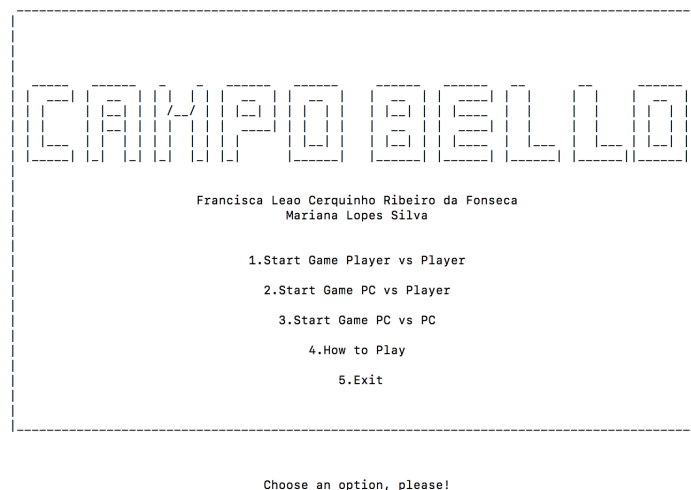


Figura 5: Menu principal

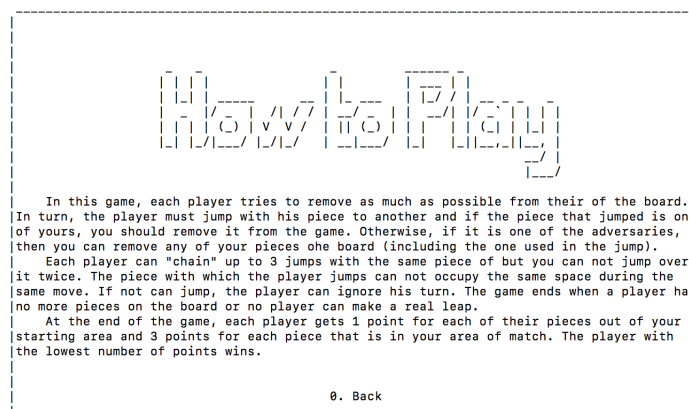


Figura 6: Menu "How To Play"

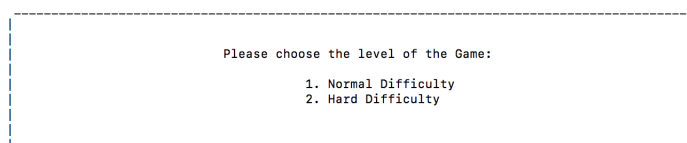


Figura 7: Menu "Set Level"

	A	B	C	D	E	F	G	H	I
1		X	X	X	X				
2			X	X	X				Y
3				X	X			Y	Y
4					N		Y	Y	Y
5	X	X	X	N	N	N	Y	Y	Y
6	X	X	X		N				
7	X	X			Y	Y			
8	X				Y	Y	Y		
9					Y	Y	Y	Y	

It is the turn of playerX

Please choose the piece that you want move:  
Please enter a position (A...I)  
|: E  
Please enter a position (1...9)  
|: 2

Row: 2 ,Col: E  
[4-5]  
What is the destiny of your piece?  
Please enter a position (A...I)  
|: E  
Please enter a position (1...9)  
|: 4

Figura 8: During the game

End Game

Points of playerX:3  
Points of playerY:0  
The winner is PlayerY

Figura 8: End Game

## 5 Conclusões

A realização deste projeto, para além de nos ser muito útil do ponto de vista algorítmico e estrutural, permitiu-nos perceber as notórias e múltiplas vantagens da linguagem *Prolog*.

Efetivamente, a solução implementada correspondeu ao que era exigido relativamente ao nível de dificuldade normal. No entanto, para o nível de dificuldade elevada a avaliação do tabuleiro poderia ter sido na totalidade implementada e com possíveis melhorias, centrando-se não só no cálculo das jogadas com o menor número de pontos, como também com as jogadas que permitissem remover o maior número de peças possível.

Em suma, este trabalho teve uma grande influência no nosso percurso como alunas de Engenharia Informática e Computação, pois permitiu-nos não só solidificar os conhecimentos lecionados, como também pôr em prática a construção dos predicados assentes nos principais pilares de *Prolog*, a unificação, a recursividade e o *backtracking*.

## 6 Bibliografia

- [1] Clocksin, W. F.; Programming in prolog. ISBN: 0-387-58350-5
- [2] *Campo Bello Game*. Acedido em Outubro e Novembro de 2017. Disponível em <http://www.campobellogame.com/blog/2016/9/27/the-campo-bello-story-tuesday-august-2-2016>
- [3] *SWI-Prolog*. Acedido em Outubro e Novembro de 2017. Disponível em <http://www.swi-prolog.org/>

## 7 Anexos

### Anexo I

#### Ficheiro "campoBello.pl"

```
1 :- include('gameLogic.pl').
2 :- include('menus.pl').
3 :- include('displayBoard.pl').
4 :- include('utilities.pl').
5 :- include('validateMoves.pl').
6 :- include('levelDifficulty.pl').
7
8 campoBello :- mainMenu.
```

### Anexo II

#### Ficheiro "displayBoard.pl"

```
1 getSymbol(empty, ' ').
2 getSymbol(pieceX1, 'X').
3 getSymbol(pieceX2, 'X').
4 getSymbol(pieceY1, 'Y').
5 getSymbol(pieceY2, 'Y').
6 getSymbol(noPiece, 'N').
7
8 initialBoard([[empty,pieceX1,pieceX1,pieceX1,pieceX1,empty,
9 empty,empty,empty],
10 [empty,empty,pieceX1,pieceX1,pieceX1,empty,empty,empty,
11 pieceY1],
12 [empty,empty,empty,pieceX1,pieceX1,empty,empty,pieceY1,
13 pieceY1],
14 [empty,empty,empty,empty,noPiece,empty,pieceY1,pieceY1,
15 pieceY1],
16 [pieceX2,pieceX2,pieceX2,noPiece,noPiece,noPiece,pieceY1,
17 pieceY1,pieceY1],
18 [pieceX2,pieceX2,pieceX2,empty,noPiece,empty,empty,empty,
19 empty],
20 [pieceX2,pieceX2,empty,empty,pieceY2,pieceY2,empty,empty,
21 empty],
22 [pieceX2,empty,empty,empty,pieceY2,pieceY2,pieceY2,empty,
23 empty],
24 [empty,empty,empty,empty,pieceY2,pieceY2,pieceY2,pieceY2,
25 empty]]).
26
27 finalBoard([[empty,noPiece,noPiece,noPiece,noPiece,empty,
28 empty,empty,empty],
29 [empty,empty,noPiece,noPiece,noPiece,empty,empty,empty,
30 noPiece],
31 [empty,empty,empty,pieceX1,noPiece,empty,empty,noPiece,
32 noPiece],
33 [empty,empty,empty,empty,pieceY1,empty,noPiece,noPiece,
34 noPiece],
35 [noPiece,noPiece,noPiece,noPiece,noPiece,noPiece,noPiece,
36 noPiece],
37 [noPiece,noPiece,pieceX2,empty,noPiece,empty,empty,empty,
38 empty],
```

```

24 [noPiece,noPiece,empty,empty,noPiece,noPiece,empty,empty,
    empty],
25 [noPiece,empty,empty,empty,empty,noPiece,noPiece,noPiece,empty,
    empty],
26 [empty,empty,empty,empty,empty,noPiece,noPiece,noPiece,noPiece,
    empty]]).
27
28 middleBoard([[empty,pieceX1,noPiece,noPiece,noPiece,empty,
    empty,empty,empty],
29 [empty,empty,noPiece,noPiece,pieceY2,empty,empty,empty,
    noPiece],
30 [empty,empty,empty,pieceX1,noPiece,empty,empty,noPiece,
    noPiece],
31 [empty,empty,empty,empty,pieceY1,empty,noPiece,noPiece,
    noPiece],
32 [noPiece,pieceX1,noPiece,noPiece,noPiece,noPiece,noPiece,
    noPiece,noPiece],
33 [noPiece,noPiece,pieceX2,empty,noPiece,empty,empty,empty,
    empty],
34 [noPiece,noPiece,empty,empty,pieceY2,noPiece,empty,empty,
    empty],
35 [noPiece,empty,empty,empty,empty,noPiece,noPiece,noPiece,empty,
    empty],
36 [empty,empty,empty,empty,empty,noPiece,noPiece,noPiece,noPiece,
    empty]]).
37
38 printFinalBoard([L|Ls]):-
39     nl,
40     printLetters,nl,
41     printBoard([L|Ls],0),
42     printLine.
43
44 printLetters:-write('      A      B      C      D      E      F
45                  G      H      I').
46
47 printSpaces:-write('      |      |      |      |      |      |
48                  |      |      |      |      |      |').
49
50 printBoard([],_).
51 printBoard([L|Ls],Y) :-
52     printLine,nl,
53     printSpaces,nl,
54     Y1 is Y+1,
55     printFinalRow(L,Y1),nl,
56     printSpaces,nl,
57     printBoard(Ls,Y1).
58
59 printFinalRow([X|Xs],Y):-
60     write(Y),
61     write(' | '), printRow([X|Xs]).
62
63 printRow([X|Xs]):-
64     getSymbol(X,Piece),
65     write(' '), write(Piece),write(' | '),
66     printRow(Xs).
67
68 printRow([]).
69 printLine:-write('

```



-----').

## Anexo III

### Ficheiro "gameLogic.pl"

```
1 :-use_module(library(lists)).
2 :-use_module(library(random)).
3 :-use_module(library(system)).
4
5 %Predicate responsible for the main game cycle
6 play(Board) :- mode_game(Curr_mode),
7   user_is(Curr_user),
8   chooseSourceCoords(RowSource, ColSource, Board, Piece,
9     AskForDestinyPiece),
9   if_then_else(AskForDestinyPiece==0,
10    chooseDestinyCoords(RowSource, ColSource, Board, Piece,
11      BoardOut),duplicate(Board,BoardOut)),nl,nl,
11   if_then_else(Curr_mode==2,
12   if_then_else(Curr_user=='pcX',set_user_is('player'),
13     set_user_is('pcX')),
13   if_then_else(Curr_mode==3,
14   if_then_else(Curr_user=='pcX',set_user_is('pcY'),set_user_is
15     ('pcX')),true)),
15   level(Curr_level),
16   write(Curr_level),
17   if_then_else(endGame(BoardOut),(nl,write('End Game'),
18     checkWinner(BoardOut)),play(BoardOut)),
18   sleep(1).
19
20 %Predicate responsible for choosing the origin coordinates
21 chooseSourceCoords(RowSource, ColSource,Board,Piece,
22   AskForDestinyPiece) :- mode_game(Curr_mode),
23   user_is(Curr_user),
24   level(Curr_level),
25   if_then_else((Curr_mode == 1; Curr_user=='player'),
26   (AskForDestinyPiece is 0,
27   repeat,
28   player(Curr_player),nl,
29   write('It is the turn of '),
30   if_then_else(Curr_mode==1,write(Curr_player),write(Curr_user
31     )),
31   nl,
32   write('Please choose the piece that you want move:'), nl,
33   write('Please enter a position (A...I)'),nl,
34   getChar(ColLetter),
35   once(letterToNumber(ColLetter, ColSource)),
36   write('Please enter a position (1...9)'),
37   nl,
38   getCode(RowSource),
39   validateSourcePiece(ColSource, RowSource,Board,Piece),
40   getPiece(Board,RowSource,ColSource,Piece)),
41   (if_then_else(Curr_level==1,
42   (if_then_else(Curr_user=='pcX',
43     listOfPiecesThatHasPossibleMoveX(FinalList,Board),
44     listOfPiecesThatHasPossibleMoveY(FinalList,Board)),
45     length(FinalList,LengthOfList),
```

```

45 if_then_else(LengthOfList==0,AskForDestinyPiece is 1,
    AskForDestinyPiece is 0),
46 random(0,LengthOfList,Index),
47 nth0(Index,FinalList,RowSource-ColSource),
48 getPiece(Board,RowSource,ColSource,Piece)),
49 (if_then_else(Curr_user=='pcX',
50 listOfPiecesThatHasPossibleMoveX(FinalList,Board),
51 listOfPiecesThatHasPossibleMoveY(FinalList,Board)),
52 listOfBestMovements(ListOfBestMoves,Board),
53 nth0(0,ListOfBestMoves,Points-RowSource-ColSource))))),
54
55 nl,write('Row: '),write(RowSource),write(' ,Col: '),
56 numberToLetter(ColSource,Letter),write(Letter),nl.
57
58 %Predicate responsible for choosing the destiny coordinates
59 chooseDestinyCoords(RowSource, ColSource, Board,Piece,
    BoardOut) :- mode_game(Curr_mode),
60 user_is(Curr_user),
61 level(Curr_level),
62 if_then_else(areaX1(RowSource,ColSource),Area='areaX1',
63 (if_then_else(areaX2(RowSource,ColSource),Area='areaX2',
64 (if_then_else(areaY1(RowSource,ColSource),Area='areaY1',
65 (if_then_else(areaY2(RowSource,ColSource),Area='areaY2',true
    ))))))),
66 listOfValidDestinyMove(List,RowSource,ColSource,Area,Board),
    length(List,LengthOfList),
67 write(List),
68 if_then_else(LengthOfList\=0,
69 (if_then_else((Curr_mode == 1; Curr_user=='player'),
70 (repeat,nl,
71 write('What is the destiny of your piece?'),
72 nl,
73 write('Please enter a position (A...I)'),
74 nl,
75 getChar(ColLetter),
76 once(letterToNumber(ColLetter, ColDestiny)),
77 write('Please enter a position (1...9)'),
78 nl,
79 getCode(RowDestiny),
80 validateDestinyPiece(ColSource,RowSource,ColDestiny,
    RowDestiny,Board,Piece,Area, BoardOut),
81 player(Curr_player),
82 if_then_else(Curr_player == 'playerX', set_player('playerY')
    ,set_player('playerX'))),
83 (if_then_else(Curr_level==1,
84 (random(0,LengthOfList,Index),
85 nth0(Index,List,RowDestiny-ColDestiny),
86 validateDestinyPiece(ColSource,RowSource,ColDestiny,
    RowDestiny,Board,Piece, Area,BoardOut)),
87 (listOfBestMovements(ListOfBestMoves,Board),
88 nth0(0,ListOfBestMoves,Points-RowDestiny-ColDestiny))))),
89 if_then_else(Curr_player == 'playerX', set_player('playerY')
    ,set_player('playerX'))),nl,
90 write('List Of Possible Moves: '),
91 write(List), write(' Row: '),write(RowDestiny), write(' Col:
    '),

```

```

92 numberToLetter(ColDestiny,Letter),write(Letter),nl.
93
94 %Predicate that returns a list with parts that have possible
    moves
95 listOfPiecesThatHasPossibleMoveX(FinalList,Board):-
96 saveElements(Board,'pieceX1',List1),
97 saveElements(Board,'pieceX2',List2),
98 append(List1,List2,ListOfDestiny),
99 scrollList(ListOfDestiny,FinalList,Board).
100
101 %Predicate that returns a list with parts that have possible
    moves
102 listOfPiecesThatHasPossibleMoveY(FinalList,BosaveElements(
    Board,'pieceY1',LsaveElements(Board,'pieceY2',Lappend(
    List1,List2,ListOfDestiny,FinalList,
    Bo%Predicate that walks through a list filling them with
    positions that have possible scrollList([],scrollList([
    Nrow-Ncol|Rest],FinalList,Board):-
103 if_then_else(areaX1(Nrow,Ncol),Area='areaX1',
104 (if_then_else(areaX2(Nrow,Ncol),Area='areaX2',
105 (if_then_else(areaY1(Nrow,Ncol),Area='areaY1',
106 (if_then_else(areaY2(Nrow,Ncol),Area='areaY2',Area='areaX1')
    ))))),
107 if_then_else(
108 % IF
109 (validateMovePC(Area,Ncol,Nrow,Col,Row,Board)),
110 % THEN
111 (scrollList(Rest,List_Temp,Board),append(List_Temp,[Nrow-
    Ncol],FinalList)),
112 % ELSE
113 scrollList(Rest,FinalList,Board)%Predicate that returns a
    list with valid target molistOfValidDestinyMove(List,
    LastRow,LastCol,Area,Board):-
114 if_then_else(setof(Nrow-Ncol,validateMovePC(Area,LastCol,
    LastRow,Ncol,Nrow,Board),List),true,
115 findall(Nrow-Ncol,validateMovePC(Area,LastCol,LastRow,Ncol,
    Nrow,Board),List)).
116
117 %Predicate that checks which pieces the player can choose to
    move
118 validateSourcePiece(Ncol,Nrow,Board,Piece):-getPiece(
    Board,Nrow,Ncol,Piece),
119 user_is(Curr_user),
120 player(Curr_player),
121 mode_game(Curr_mode),
122 if_then_else(Curr_mode==1,if_then_else(Curr_player=='playerX
    ',
123 (Piece \= 'pieceY1',
124 Piece \= 'pieceY2'),
125 (Piece \= 'pieceX1',
126 Piece \= 'pieceX2')),
127 if_then_else(Curr_user=='pcX',
128 (Piece \= 'pieceY1',
129 Piece \= 'pieceY2'),
130 (Piece \= 'pieceX1',
131 Piece \= 'pieceX2'))),

```

```

132 Piece \= 'empty',
133 Piece \= 'noPiece'.
134
135 %Predicate that verifies the valid movements
136 validateDestinyPiece(LastCol,LastRow,Ncol,Nrow,Board, Piece,
    Area,BoardOut) :- if_then_else((Piece=='pieceX1';Piece=='
    pieceX2'),
137 checkIfCanMoveX(Ncol, Nrow, LastCol,LastRow,Board,Piece,
    BoardOut,Area),
138 checkIfCanMoveY(Ncol, Nrow, LastCol,LastRow,Board,Piece,
    BoardOut,Area)).
139
140 %Predicate that the piece to which it jumped is a noPiece
    and in case it is asked to jump again.
141 checkIfIsNotNoPiece(Board,BoardOut,LastColPiece,LastRowPiece
    ,Row,Col,FinalRow,FinalCol,Piece,Area,NewContinue):-
    repeat,
142 chooseNewJump(Board,BoardOut,LastColPiece,LastRowPiece,Row,
    Col,FinalRow,FinalCol,Piece,Area,NewContinue),
143 if_then_else(NewContinue\=1,
144 (getPiece(BoardOut, FinalRow, FinalCol, SecondPiece),
145 SecondPiece \= 'noPiece'),true).
146
147 %Predicate that calls the function responsible for printing
    the board after a new jump.
148 printBoardAfterJump(Row,Col,LastRow,LastCol,Board,BoardOut,
    Piece) :- setPiece(Board,LastRow,LastCol,'noPiece',
    BoardOut2),
149 setPiece(BoardOut2,Row,Col,Piece,BoardOut),
150 printFinalBoard(BoardOut),nl.
151
152 %Predicate that checks if it did not jump to the same piece
    in the same movement.
153 checkIfIsNotRedo(LastColPiece,LastRowPiece,ColPiece,RowPiece
    ):-LastColPiece==ColPiece,
154 LastRowPiece==RowPiece.
155
156 %Predicate responsible for executing all rules of the game
    for player X, namely when the user can make single,
    double or triple jumps, validating them.
157 checkIfCanMoveX(Ncol,Nrow,LastCol,LastRow,Board,Piece,
    BoardOut,Area) :-
158 validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
159 getPiece(Board, Nrow, Ncol, NewPiece),
160 if_then_else((NewPiece=='noPiece'),
161 (printBoardAfterJump(Nrow,Ncol,LastRow,LastCol,Board,
    BoardOut2,Piece),(chooseNewJump(BoardOut2,BoardOut3,
    LastCol,LastRow,Nrow, Ncol,Row,Col,Piece,Area,Continue),
162 if_then_else(Continue\=1,(getPiece(Board,Row,Col,Piece2),
163 if_then_else(areaX1(Row,Col),Area2='areaX1',
164 (if_then_else(areaX2(Row,Col),Area2='areaX2',
165 (if_then_else(areaY1(Row,Col),Area2='areaY1',
166 (if_then_else(areaY2(Row,Col),Area2='areaY2',Area2=Area))))
    )),
167 if_then_else(Piece2=='noPiece',(checkIfIsNotNoPiece(
    BoardOut3,BoardOut4,Ncol,Nrow,Row,Col,FinalRow,FinalCol,

```

```

        SecondPiece,Area2,NewContinue),
168 if_then_else(NewContinue\=1,
169 (if_then_else(areaX1(FinalRow,FinalCol),Area3='areaX1',
170 (if_then_else(areaX2(FinalRow,FinalCol),Area3='areaX2',
171 (if_then_else(areaY1(FinalRow,FinalCol),Area3='areaY1',
172 (if_then_else(areaY2(FinalRow,FinalCol),Area3='areaY2',Area3
    =Area2)))))),
173 (if_then_else((SecondPiece=='pieceY1';SecondPiece=='pieceY2'
    ),
174 (choosePieceToRemove(BoardOut4, BoardOut5),
175 setPiece(BoardOut5,FinalRow,FinalCol,Piece,BoardOut6),
    setPiece(BoardOut6,FinalRow,FinalCol,'noPiece',BoardOut))
    ,(validateMove(Area3, Col, Row, FinalCol, FinalRow,
    BoardOut4),
176 setPiece(BoardOut4,FinalRow,FinalCol,Piece,BoardOut))))),
    duplicate(BoardOut4,BoardOut))),
177 (if_then_else((Piece2=='pieceY1';Piece2=='pieceY2'),
178 (choosePieceToRemove(BoardOut3, BoardOut4),
179 setPiece(BoardOut4,Row,Col,Piece,BoardOut)),(setPiece(
    BoardOut3,Row,Col,Piece,BoardOut))))),duplicate(
    BoardOut3,BoardOut))),
180 (if_then_else((NewPiece=='pieceY1';NewPiece=='pieceY2'), (
    validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
181 choosePieceToRemove(Board, BoardOut2),setPiece(BoardOut2,
    LastRow,LastCol,'noPiece',BoardOut3),setPiece(BoardOut3,
    Nrow,Ncol,Piece,BoardOut)),
182 (validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
    setPiece(Board,LastRow,LastCol,'noPiece',BoardOut2),
183 setPiece(BoardOut2,Nrow,Ncol,Piece,BoardOut))))),
184 printFinalBoard(BoardOut).
185
186 %Predicate responsible for executing all rules of the game
    for player Y, namely when the user can make single,
    double or triple jumps, validating them.
187 checkIfCanMoveY(Ncol,Nrow,LastCol,LastRow,Board,Piece,
    BoardOut,Area) :-
188 validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
189 getPiece(Board, Nrow, Ncol, NewPiece),
190 if_then_else((NewPiece=='noPiece'),
191 (printBoardAfterJump(Nrow,Ncol,LastRow,LastCol,Board,
    BoardOut2,Piece),(chooseNewJump(BoardOut2,BoardOut3,
    LastCol,LastRow,Nrow, Ncol,Row,Col,Piece,Area,Continue),
192 if_then_else(Continue\=1,(getPiece(Board,Row,Col,Piece2),
193 if_then_else(areaX1(Row,Col),Area2='areaX1',
194 (if_then_else(areaX2(Row,Col),Area2='areaX2',
195 (if_then_else(areaY1(Row,Col),Area2='areaY1',
196 (if_then_else(areaY2(Row,Col),Area2='areaY2',Area2=Area))))))
    )),
197 if_then_else(Piece2=='noPiece',(checkIfIsNotNoPiece(
    BoardOut3,BoardOut4,Ncol,Nrow,Row,Col,FinalRow,FinalCol,
    SecondPiece,Area2,NewContinue),
198 if_then_else(NewContinue\=1,
199 (if_then_else(areaX1(FinalRow,FinalCol),Area3='areaX1',
200 (if_then_else(areaX2(FinalRow,FinalCol),Area3='areaX2',
201 (if_then_else(areaY1(FinalRow,FinalCol),Area3='areaY1',
202 (if_then_else(areaY2(FinalRow,FinalCol),Area3='areaY2',Area3

```

```

=Area2)))))),
203 (if_then_else((SecondPiece=='pieceX1';SecondPiece=='pieceX2'
),
204 (choosePieceToRemove(BoardOut4, BoardOut5),
205 setPiece(BoardOut5,FinalRow,FinalCol,Piece,BoardOut6),
setPiece(BoardOut6,FinalRow,FinalCol,'noPiece',BoardOut))
,(validateMove(Area3, Col, Row, FinalCol, FinalRow,
BoardOut4),
206 setPiece(BoardOut4,FinalRow,FinalCol,Piece,BoardOut))))),
duplicate(BoardOut3,BoardOut))),
207 (if_then_else((Piece2=='pieceX1';Piece2=='pieceX2'),
208 (choosePieceToRemove(BoardOut3, BoardOut4),
209 setPiece(BoardOut4,Row,Col,Piece,BoardOut)),(setPiece(
BoardOut3,Row,Col,Piece,BoardOut))))),duplicate(
BoardOut2,BoardOut))),
210 (if_then_else((NewPiece=='pieceX1';NewPiece=='pieceX2'), (
validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
211 choosePieceToRemove(Board, BoardOut2),setPiece(BoardOut2,
LastRow,LastCol,'noPiece',BoardOut3),setPiece(BoardOut3,
Nrow,Ncol,Piece,BoardOut)),
212 (validateMove(Area, LastCol, LastRow, Ncol, Nrow,Board),
setPiece(Board,LastRow,LastCol,'noPiece',BoardOut2),
213 setPiece(BoardOut2,Nrow,Ncol,Piece,BoardOut))))),
214 printFinalBoard(BoardOut).
215
216 %Predicate responsible for requesting a new leap to the
player when a "noPiece" part is found
217 chooseNewJump(Board,BoardOut,LastColPiece,LastRowPiece,
LastRow,LastCol,Row,Col,Piece,Area,Continue) :-
218 if_then_else(areaX1(LastRow,LastCol),AreaDestiny='areaX1',
219 (if_then_else(areaX2(LastRow,LastCol),AreaDestiny='areaX2',
220 (if_then_else(areaY1(LastRow,LastCol),AreaDestiny='areaY1',
221 (if_then_else(areaY2(LastRow,LastCol),AreaDestiny='areaY2',
AreaDestiny=Area)))))),
222 mode_game(Curr_mode),
223 user_is(Curr_user),
224 listOfValidDestinyMove(List,LastRow,LastCol,AreaDestiny,
Board),length(List,LengthOfList),
225 if_then_else(LengthOfList\=0,
226 if_then_else((Curr_mode == 1; Curr_user=='player'),
227 (repeat,nl,write('You need jump one more time!'),
228 nl,
229 write('Please enter a position (A...I)'),
230 nl,
231 getChar(ColLetter),
232 once(letterToNumber(ColLetter, Col)),
233 write('Please enter a position (1...9)'),
234 nl,
235 getCode(Row),
236 if_then_else(checkIfIsNotRedo(LastColPiece,LastRowPiece,Col,
Row),
237 if_then_else(LengthOfList==1,(write('Cant move to this Piece
'),
238 Continue is 1,setPiece(Board,LastRowPiece,LastColPiece,Piece
,BoardOut2),
239 getPiece(Board,Row,Col,PieceChoosen),setPiece(BoardOut2,Row,

```

```

        Col,PieceChooosen,BoardOut)),
240 (Continue is 0,
241 chooseNewJump(Board,BoardOut,LastColPiece,LastRowPiece,
        LastRow,LastCol,Row,Col,Piece,Area,Continue))),
242 (Continue is 0,
243 validateMove(AreaDestiny, LastCol, LastRow, Col, Row,Board),
244 setPiece(Board,Row,Col,Piece,BoardOut2),
245 setPiece(BoardOut2,LastRow,LastCol,'noPiece',BoardOut),
246 (nl,write('List Of Possible Moves: '),write(List),
247 write(' Row: '),write(Row),write(' Col: '),write(Col),
248 printFinalBoard(BoardOut))))) ,
249 (nl,write('Row: '),write(LastRow),write(' Col: '),write(
        LastCol),nl,
250 nl,write('Lista: '),write(List),
251 if_then_else(checkIfIsNotRedo(LastColPiece,LastRowPiece,Col,
        Row),
252 if_then_else(LengthOfList==1,(write('Cant move to this Piece
        ')),
253 Continue is 1,duplicate(Board,BoardOut),
254 printFinalBoard(BoardOut)),
255 (Continue is 0, random(0,LengthOfList,Index),
256 nth0(Index,List,Row-Col),
257 setPiece(Board,Row,Col,Piece,BoardOut2),
258 setPiece(BoardOut2,LastRow,LastCol,'noPiece',BoardOut),
259 nl,write('List Of Possible Moves: '),write(List),
260 write(' Row: '),write(Row),write(' Col: '),write(Col),
261 printFinalBoard(BoardOut))),
262 (Continue is 0,
263 random(0,LengthOfList,Index),
264 nth0(Index,List,Row-Col),
265 setPiece(Board,Row,Col,Piece,BoardOut2),
266 setPiece(BoardOut2,LastRow,LastCol,'noPiece',BoardOut),
267 nl,write('List Of Possible Moves: '),write(List),
268 write(' Row: '),write(Row),write(' Col: '),write(Col),
269 printFinalBoard(BoardOut))))) ,
270 (write('Without Possible Moves!')),
271 duplicate(Board,BoardOut),
272 printFinalBoard(BoardOut))).
273
274
275
276 %Predicate responsible for asking the user which part of the
        board he wants to remove
277 choosePieceToRemove(Board, BoardOut) :-mode_game(Curr_mode),
278 user_is(Curr_user),
279 player(Curr_player),
280 if_then_else((Curr_mode == 1; Curr_user == 'player'),
281 (repeat,nl, write('What is the piece that you want remove?')
        ,
282 nl,
283 write('Please enter a position (A...I)'),
284 nl,
285 getChar(ColLetter),
286 once(letterToNumber(ColLetter, Col)),
287 write('Please enter a position (1...9)'),
288 nl,

```

```

289 getCode(Row),
290 if_then_else(Curr_mode==1,
291 if_then_else(Curr_player=='playerX',
292 checkIfCanRemoveX(Board, Col, Row),
293 checkIfCanRemoveY(Board, Col, Row)),
294 if_then_else(Curr_user=='pcX',
295 checkIfCanRemoveX(Board, Col, Row),
296 checkIfCanRemoveY(Board, Col, Row)))),
297 (if_then_else(Curr_user=='pcX',
298 listOfPiecesThatCanRemoveX(Board,List),
299 listOfPiecesThatCanRemoveY(Board,List)),
300 length(List,LengthOfList),
301 random(0,LengthOfList,Index),
302 nth0(Index,List,Row-Col))),
303 setPiece(Board,Row,Col,'noPiece',BoardOut).
304
305 %Predicate that returns a list of the parts that the X
    computer can remove
306 listOfPiecesThatCanRemoveX(Board,List):-if_then_else(setof(
    Nrow-Ncol,checkIfCanRemoveX(Board,Ncol,Nrow),List),true,
307 findall(Nrow-Ncol,checkIfCanRemoveX(Board,Ncol,Nrow),List)).
308
309 %Predicate that returns a list of the parts that the X
    computer can remove
310 listOfPiecesThatCanRemoveY(Board,List):-if_then_else(setof(
    Nrow-Ncol,checkIfCanRemoveY(Board,Ncol,Nrow),List),true,
311 findall(Nrow-Ncol,checkIfCanRemoveY(Board,Ncol,Nrow),List)).
312
313 %Predicate that tests whether the piece chosen by player X
    to remove is one of his own pieces.
314 checkIfCanRemoveX(Board, Col, Row) :- getPiece(Board, Row,
    Col, NewPiece),
315 NewPiece \= 'empty',
316 NewPiece \= 'pieceY1',
317 NewPiece \= 'pieceY2',
318 NewPiece \= 'noPiece'.
319
320 %Predicate that tests whether the piece chosen by player X
    to remove is one of his own pieces.
321 checkIfCanRemoveY(Board, Col, Row) :- getPiece(Board, Row,
    Col, NewPiece),
322 NewPiece \= 'empty',
323 NewPiece \= 'pieceX1',
324 NewPiece \= 'pieceX2',
325 NewPiece \= 'noPiece'.
326
327 %Predicate that returns the part in a given row and column.
328 getPiece(Board, Nrow, Ncol, Piece) :- getElement(Board,Nrow
    ,Ncol,Piece).
329
330 %Predicate that changes a certain piece in the board and
    updates the new board.
331 setPiece(BoardIn, Nrow, Ncol, Piece, BoardOut) :- setOnRow(
    Nrow, BoardIn, Ncol, Piece, BoardOut).
332
333 %Predicate that changes the part on the board in a certain

```



```

line
334 setOnRow(1, [Row|Remainder], Ncol, Piece, [Newrow|Remainder
    ]):- setOnCol(Ncol, Row, Piece, Newrow).
335 setOnRow(Pos, [Row|Remainder], Ncol, Piece, [Row|
    Newremainder]):- Pos @> 1,
336 Next is Pos-1,
337 setOnRow(Next, Remainder, Ncol, Piece, Newremainder).
338 %Predicate that changes the part on the board in a given
    column
339 setOnCol(1, [_|Remainder], Piece, [Piece|Remainder]).
340 setOnCol(Pos, [X|Remainder], Piece, [X|Newremainder]):- Pos
    @> 1,
341 Next is Pos-1,
342 setOnCol(Next, Remainder, Piece, Newremainder).
343 %Predicate that does an if than else
344 if_then_else(If, Then,_):- If,!, Then.
345 if_then_else(_, _, Else):- Else.
346
347 %Predicate that returns the element that is contained in the
    tray in a given column and row.
348 getElement(Board,Nrow,Ncol,Element) :- nth1(Nrow, Board, Row
    ),
349 nth1(Ncol,Row,Element).
350
351 %Predicate that checks if is in area X1
352 areaX1(Nrow,Ncol):- (Ncol@>1,
353 Ncol@<6,
354 Nrow@>0,
355 Nrow@<5).
356 %Predicate that checks if is in area X2
357 areaX2(Nrow,Ncol):- (Ncol@>0,
358 Ncol@<5,
359 Nrow@>4,
360 Nrow@<9).
361
362 %Predicate that checks if is in area Y1
363 areaY1(Nrow,Ncol):- (Ncol@>5,
364 Ncol@<10,
365 Nrow@>1,
366 Nrow@<6).
367
368 %Predicate that checks if is in area Y2
369 areaY2(Nrow,Ncol):- (Ncol@>4,
370 Ncol@<9,
371 Nrow@>5,
372 Nrow@<10).
373
374 %Predicate that checks if is the area of playerX
375 areaX(Nrow,Ncol):- (Ncol@>1,
376 Ncol@<6,
377 Nrow@>0,
378 Nrow@<5);
379 (Ncol@>0,
380 Ncol@<5,
381 Nrow@>4,
382 Nrow@<9).

```

```

383
384 %Predicate that checks if is the area of playerY
385 areaY(Nrow,Ncol):- (Ncol@>5,
386 Ncol@<10,
387 Nrow@>1,
388 Nrow@<6);
389 (Ncol@>4,
390 Ncol@<9,
391 Nrow@>5,
392 Nrow@<10).
393
394 %Predicate that does a list with the pieces of the player
395 saveElements(Board,'pieceX1',List):- if_then_else(setof(Nrow
-Ncol,getElement(Board,Nrow,Ncol,'pieceX1'),List),
396 true,findall(Nrow-Ncol,getElement(Board,Nrow,Ncol,'pieceX1')
,List)).
397 saveElements(Board,'pieceX2',List):- if_then_else(setof(Nrow
-Ncol,getElement(Board,Nrow,Ncol,'pieceX2'),List),
398 true,findall(Nrow-Ncol,getElement(Board,Nrow,Ncol,'pieceX2')
,List)).
399 saveElements(Board,'pieceY1',List):- if_then_else(setof(Nrow
-Ncol,getElement(Board,Nrow,Ncol,'pieceY1'),List),
400 true,findall(Nrow-Ncol,getElement(Board,Nrow,Ncol,'pieceY1')
,List)).
401 saveElements(Board,'pieceY2',List):- if_then_else(setof(Nrow
-Ncol,getElement(Board,Nrow,Ncol,'pieceY2'),List),
402 true,findall(Nrow-Ncol,getElement(Board,Nrow,Ncol,'pieceY2')
,List)).
403
404 %Predicatat that checks which area the piece is in and
calculate the points
405 getNrowNcol([],PointsXIn,PointsXOut,'playerX').
406 getNrowNcol([],PointsYIn,PointsYOut,'playerY').
407 getNrowNcol([Nrow-Ncol|Rest],PointsXIn,PointsXOut,'playerX')
:-
408 if_then_else(areaX(Nrow,Ncol),PointsXOut is PointsXIn+3,
409 PointsXOut is PointsXIn+1),nl,
410 getNrowNcol(Rest,PointsXOut,PointsXOutNew,'playerX').
411 getNrowNcol([Nrow-Ncol|Rest],PointsYIn,PointsYOut,'playerY')
:-
412 if_then_else(areaY(Nrow,Ncol),PointsYOut is PointsYIn+3,
413 PointsYOut is PointsYIn+1),
414 getNrowNcol(Rest,PointsYOut,PointsYOutNew,'playerY').
415
416 %Predicate that checks if the playerX has pieces on the
board
417 checkIfExistsPiecesX(Board) :- saveElements(Board,'pieceX1',
List),
418 saveElements(Board,'pieceX2',List2),
419 append(List,List2,FinalList),
420 length(FinalList,LengthOfFinalList),
421 if_then_else(LengthOfFinalList==0,fail,true).
422
423 %Predicate that checks if the playerY has pieces on the
board
424 checkIfExistsPiecesY(Board) :- saveElements(Board,'pieceY1',

```

```

    ,List),
425 saveElements(Board,'pieceY2',List2),
426 append(List,List2,FinalList),
427 length(FinalList,LengthOfFinalList),
428 if_then_else(LengthOfFinalList==0,fail,true).
429
430 %Predicate that checks if is the end of the game
431 endGame(Board) :- listOfPiecesThatHasPossibleMoveX(FinalList
    ,Board),
432 length(FinalList,LengthOfFinalList),
433 listOfPiecesThatHasPossibleMoveY(FinalList2,Board),
434 length(FinalList2,LengthOfFinalList2),
435 (if_then_else(checkIfExistsPiecesY(Board),fail,true);
436 if_then_else(checkIfExistsPiecesX(Board),fail,true);
437 if_then_else(LengthOfFinalList==0,true,fail);
438 if_then_else(LengthOfFinalList2==0,true,fail)).
439
440 %Predicate that calculates the points of each player
441 calculatePoints(Board,PointsX,PointsY):- saveElements(Board,
    'pieceX1',List),
442 saveElements(Board,'pieceX2',List2),
443 append(List,List2,FinalListX),
444 getNrowNcol(FinalListX,0,PointsX,'playerX'),
445 length(FinalListX,LengthOfFinalListX),
446 if_then_else(LengthOfFinalListX==0,PointsX is 0,true),
447 saveElements(Board,'pieceY1',List3),
448 saveElements(Board,'pieceY2',List4),
449 append(List3,List4,FinalListY),
450 getNrowNcol(FinalListY,0,PointsY,'playerY'),
451 length(FinalListY,LengthOfFinalListY),
452 if_then_else(LengthOfFinalListY==0,PointsY is 0,true),
453 nl,
454 write('Points of playerX:'), write(PointsX),nl,
455 write('Points of playerY:'), write(PointsY),nl.
456
457 %Predicate that checks the winner of the game
458 checkWinner(Board) :- calculatePoints(Board,PointsX,PointsY)
    ,
459 if_then_else(PointsX@>PointsY,write('The winner is PlayerY')
    ,write('The winner is PlayerX')).

```

## Anexo IV

### Ficheiro "levelDisfficulty.pl"

```

1 %Predicate that calculates the player's points through the
    received board
2 evaluateBoards(Board,Points):-user_is(Curr_user),
3 if_then_else(Curr_user=='pcX',
4 (saveElements(Board,'pieceX1',List),
5 saveElements(Board,'pieceX2',List2),
6 append(List,List2,FinalList),
7 getNrowNcol(FinalList,0,Points,'playerX')),
8
9 (saveElements(Board,'pieceY1',List),
10 saveElements(Board,'pieceY2',List2),
11 append(List,List2,FinalList),

```

```

12 getNrowNcol(FinalList,0,Points,'playerY'))).
13
14 %Instance Nrow e Ncol
15 is_member([],_,_).
16 is_member([Nrow-Ncol|Rest],Row,Col):-if_then_else((Row ==
    Nrow,Col == Ncol),true,is_member(Rest,Row,Col)).
17
18 %Predicate that makes a list with the best plays,
    increasingly ordered by the number of points of each
    player
19 listOfBestMovements(FinalList,Board):-
20     user_is(Curr_user),
21     setof(Points-Nrow-Ncol,(
22         if_then_else(Curr_user=='pcX',
23             listOfPiecesThatHasPossibleMoveX(List,Board),
24             listOfPiecesThatHasPossibleMoveY(List,Board)),
25         is_member(List,Nrow,Ncol),
26         if_then_else(areaX1(Nrow,Ncol),Area='areaX1',
27             (if_then_else(areaX2(Nrow,Ncol),Area='areaX2',
28                 (if_then_else(areaY1(Nrow,Ncol),Area='areaY1',
29                     (if_then_else(areaY2(Nrow,Ncol),Area='areaY2',Area='areaX1
30                         '))))))),write(Nrow),write(Ncol),
31         validateMovePC(Area,Ncol,Nrow,Col,Row,Board),getPiece(
32             Board,Nrow,Ncol,Piece),
33         setPiece(Board,Col,Row,Piece,BoardOut2),setPiece(BoardOut2
34             ,Nrow,Ncol,'noPiece',BoardOut),
35         evaluateBoards(BoardOut,Points)),FinalList).

```

## Anexo V

### Ficheiro "menus.pl"

```

1 printMainMenu:-
2     nl,nl,nl,
3     write('
4         -----
5         '),nl,
6         write('          |
7
8         |'),nl,
9         write('          |
10
11         |'),nl,
12         write('          |
13
14         |'),nl,
15         write('          |
16
17         |'),nl,
18         write('          |
19
20         |'),nl,
21         write('          |
22
23         |'),nl,
24         write('          |
25
26         |'),nl,
27         write('          |
28
29         |'),nl,
30         write('          |
31
32         |'),nl,
33         write('          |
34
35         |'),nl,
36         write('          |
37
38         |'),nl,
39         write('          |
40
41         |'),nl,
42         write('          |
43
44         |'),nl,
45         write('          |
46
47         |'),nl,
48         write('          |
49
50         |'),nl,
51         write('          |
52
53         |'),nl,
54         write('          |
55
56         |'),nl,
57         write('          |
58
59         |'),nl,
60         write('          |
61
62         |'),nl,
63         write('          |
64
65         |'),nl,
66         write('          |
67
68         |'),nl,
69         write('          |
70
71         |'),nl,
72         write('          |
73
74         |'),nl,
75         write('          |
76
77         |'),nl,
78         write('          |
79
80         |'),nl,
81         write('          |
82
83         |'),nl,
84         write('          |
85
86         |'),nl,
87         write('          |
88
89         |'),nl,
90         write('          |
91
92         |'),nl,
93         write('          |
94
95         |'),nl,
96         write('          |
97
98         |'),nl,
99         write('          |
100
101         |'),nl,
102         write('          |
103
104         |'),nl,
105         write('          |
106
107         |'),nl,
108         write('          |
109
110         |'),nl,
111         write('          |
112
113         |'),nl,
114         write('          |
115
116         |'),nl,
117         write('          |
118
119         |'),nl,
120         write('          |
121
122         |'),nl,
123         write('          |
124
125         |'),nl,
126         write('          |
127
128         |'),nl,
129         write('          |
130
131         |'),nl,
132         write('          |
133
134         |'),nl,
135         write('          |
136
137         |'),nl,
138         write('          |
139
140         |'),nl,
141         write('          |
142
143         |'),nl,
144         write('          |
145
146         |'),nl,
147         write('          |
148
149         |'),nl,
150         write('          |
151
152         |'),nl,
153         write('          |
154
155         |'),nl,
156         write('          |
157
158         |'),nl,
159         write('          |
160
161         |'),nl,
162         write('          |
163
164         |'),nl,
165         write('          |
166
167         |'),nl,
168         write('          |
169
170         |'),nl,
171         write('          |
172
173         |'),nl,
174         write('          |
175
176         |'),nl,
177         write('          |
178
179         |'),nl,
180         write('          |
181
182         |'),nl,
183         write('          |
184
185         |'),nl,
186         write('          |
187
188         |'),nl,
189         write('          |
190
191         |'),nl,
192         write('          |
193
194         |'),nl,
195         write('          |
196
197         |'),nl,
198         write('          |
199
200         |'),nl,
201         write('          |
202
203         |'),nl,
204         write('          |
205
206         |'),nl,
207         write('          |
208
209         |'),nl,
210         write('          |
211
212         |'),nl,
213         write('          |
214
215         |'),nl,
216         write('          |
217
218         |'),nl,
219         write('          |
220
221         |'),nl,
222         write('          |
223
224         |'),nl,
225         write('          |
226
227         |'),nl,
228         write('          |
229
230         |'),nl,
231         write('          |
232
233         |'),nl,
234         write('          |
235
236         |'),nl,
237         write('          |
238
239         |'),nl,
240         write('          |
241
242         |'),nl,
243         write('          |
244
245         |'),nl,
246         write('          |
247
248         |'),nl,
249         write('          |
250
251         |'),nl,
252         write('          |
253
254         |'),nl,
255         write('          |
256
257         |'),nl,
258         write('          |
259
260         |'),nl,
261         write('          |
262
263         |'),nl,
264         write('          |
265
266         |'),nl,
267         write('          |
268
269         |'),nl,
270         write('          |
271
272         |'),nl,
273         write('          |
274
275         |'),nl,
276         write('          |
277
278         |'),nl,
279         write('          |
280
281         |'),nl,
282         write('          |
283
284         |'),nl,
285         write('          |
286
287         |'),nl,
288         write('          |
289
290         |'),nl,
291         write('          |
292
293         |'),nl,
294         write('          |
295
296         |'),nl,
297         write('          |
298
299         |'),nl,
300         write('          |
301
302         |'),nl,
303         write('          |
304
305         |'),nl,
306         write('          |
307
308         |'),nl,
309         write('          |
310
311         |'),nl,
312         write('          |
313
314         |'),nl,
315         write('          |
316
317         |'),nl,
318         write('          |
319
320         |'),nl,
321         write('          |
322
323         |'),nl,
324         write('          |
325
326         |'),nl,
327         write('          |
328
329         |'),nl,
330         write('          |
331
332         |'),nl,
333         write('          |
334
335         |'),nl,
336         write('          |
337
338         |'),nl,
339         write('          |
340
341         |'),nl,
342         write('          |
343
344         |'),nl,
345         write('          |
346
347         |'),nl,
348         write('          |
349
350         |'),nl,
351         write('          |
352
353         |'),nl,
354         write('          |
355
356         |'),nl,
357         write('          |
358
359         |'),nl,
360         write('          |
361
362         |'),nl,
363         write('          |
364
365         |'),nl,
366         write('          |
367
368         |'),nl,
369         write('          |
370
371         |'),nl,
372         write('          |
373
374         |'),nl,
375         write('          |
376
377         |'),nl,
378         write('          |
379
380         |'),nl,
381         write('          |
382
383         |'),nl,
384         write('          |
385
386         |'),nl,
387         write('          |
388
389         |'),nl,
390         write('          |
391
392         |'),nl,
393         write('          |
394
395         |'),nl,
396         write('          |
397
398         |'),nl,
399         write('          |
400
401         |'),nl,
402         write('          |
403
404         |'),nl,
405         write('          |
406
407         |'),nl,
408         write('          |
409
410         |'),nl,
411         write('          |
412
413         |'),nl,
414         write('          |
415
416         |'),nl,
417         write('          |
418
419         |'),nl,
420         write('          |
421
422         |'),nl,
423         write('          |
424
425         |'),nl,
426         write('          |
427
428         |'),nl,
429         write('          |
430
431         |'),nl,
432         write('          |
433
434         |'),nl,
435         write('          |
436
437         |'),nl,
438         write('          |
439
440         |'),nl,
441         write('          |
442
443         |'),nl,
444         write('          |
445
446         |'),nl,
447         write('          |
448
449         |'),nl,
450         write('          |
451
452         |'),nl,
453         write('          |
454
455         |'),nl,
456         write('          |
457
458         |'),nl,
459         write('          |
460
461         |'),nl,
462         write('          |
463
464         |'),nl,
465         write('          |
466
467         |'),nl,
468         write('          |
469
470         |'),nl,
471         write('          |
472
473         |'),nl,
474         write('          |
475
476         |'),nl,
477         write('          |
478
479         |'),nl,
480         write('          |
481
482         |'),nl,
483         write('          |
484
485         |'),nl,
486         write('          |
487
488         |'),nl,
489         write('          |
490
491         |'),nl,
492         write('          |
493
494         |'),nl,
495         write('          |
496
497         |'),nl,
498         write('          |
499
500         |'),nl,
501         write('          |
502
503         |'),nl,
504         write('          |
505
506         |'),nl,
507         write('          |
508
509         |'),nl,
509         write('          |
510
511         |'),nl,
512         write('          |
513
514         |'),nl,
515         write('          |
516
517         |'),nl,
518         write('          |
519
520         |'),nl,
521         write('          |
522
523         |'),nl,
524         write('          |
525
526         |'),nl,
527         write('          |
528
529         |'),nl,
530         write('          |
531
532         |'),nl,
533         write('          |
534
535         |'),nl,
536         write('          |
537
538         |'),nl,
539         write('          |
540
541         |'),nl,
542         write('          |
543
544         |'),nl,
545         write('          |
546
547         |'),nl,
548         write('          |
549
550         |'),nl,
551         write('          |
552
553         |'),nl,
554         write('          |
555
556         |'),nl,
557         write('          |
558
559         |'),nl,
560         write('          |
561
562         |'),nl,
563         write('          |
564
565         |'),nl,
566         write('          |
567
568         |'),nl,
569         write('          |
570
571         |'),nl,
572         write('          |
573
574         |'),nl,
575         write('          |
576
577         |'),nl,
578         write('          |
579
580         |'),nl,
581         write('          |
582
583         |'),nl,
584         write('          |
585
586         |'),nl,
587         write('          |
588
589         |'),nl,
590         write('          |
591
592         |'),nl,
593         write('          |
594
595         |'),nl,
596         write('          |
597
598         |'),nl,
599         write('          |
600
601         |'),nl,
602         write('          |
603
604         |'),nl,
605         write('          |
606
607         |'),nl,
608         write('          |
609
610         |'),nl,
611         write('          |
612
613         |'),nl,
614         write('          |
615
616         |'),nl,
617         write('          |
618
619         |'),nl,
620         write('          |
621
622         |'),nl,
623         write('          |
624
625         |'),nl,
626         write('          |
627
628         |'),nl,
629         write('          |
630
631         |'),nl,
632         write('          |
633
634         |'),nl,
635         write('          |
636
637         |'),nl,
638         write('          |
639
640         |'),nl,
641         write('          |
642
643         |'),nl,
644         write('          |
645
646         |'),nl,
647         write('          |
648
649         |'),nl,
650         write('          |
651
652         |'),nl,
653         write('          |
654
655         |'),nl,
656         write('          |
657
658         |'),nl,
659         write('          |
660
661         |'),nl,
662         write('          |
663
664         |'),nl,
665         write('          |
666
667         |'),nl,
668         write('          |
669
670         |'),nl,
671         write('          |
672
673         |'),nl,
674         write('          |
675
676         |'),nl,
677         write('          |
678
679         |'),nl,
680         write('          |
681
682         |'),nl,
683         write('          |
684
685         |'),nl,
686         write('          |
687
688         |'),nl,
689         write('          |
690
691         |'),nl,
692         write('          |
693
694         |'),nl,
695         write('          |
696
697         |'),nl,
698         write('          |
699
700         |'),nl,
701         write('          |
702
703         |'),nl,
704         write('          |
705
706         |'),nl,
707         write('          |
708
709         |'),nl,
710         write('          |
711
712         |'),nl,
713         write('          |
714
715         |'),nl,
716         write('          |
717
718         |'),nl,
719         write('          |
720
721         |'),nl,
722         write('          |
723
724         |'),nl,
725         write('          |
726
727         |'),nl,
728         write('          |
729
730         |'),nl,
731         write('          |
732
733         |'),nl,
734         write('          |
735
736         |'),nl,
737         write('          |
738
739         |'),nl,
740         write('          |
741
742         |'),nl,
743         write('          |
744
745         |'),nl,
746         write('          |
747
748         |'),nl,
749         write('          |
750
751         |'),nl,
752         write('          |
753
754         |'),nl,
755         write('          |
756
757         |'),nl,
758         write('          |
759
760         |'),nl,
761         write('          |
762
763         |'),nl,
764         write('          |
765
766         |'),nl,
767         write('          |
768
769         |'),nl,
770         write('          |
771
772         |'),nl,
773         write('          |
774
775         |'),nl,
776         write('          |
777
778         |'),nl,
779         write('          |
780
781         |'),nl,
782         write('          |
783
784         |'),nl,
785         write('          |
786
787         |'),nl,
788         write('          |
789
790         |'),nl,
791         write('          |
792
793         |'),nl,
794         write('          |
795
796         |'),nl,
797         write('          |
798
799         |'),nl,
800         write('          |
801
802         |'),nl,
803         write('          |
804
805         |'),nl,
806         write('          |
807
808         |'),nl,
809         write('          |
810
811         |'),nl,
812         write('          |
813
814         |'),nl,
815         write('          |
816
817         |'),nl,
818         write('          |
819
820         |'),nl,
821         write('          |
822
823         |'),nl,
824         write('          |
825
826         |'),nl,
827         write('          |
828
829         |'),nl,
830         write('          |
831
832         |'),nl,
833         write('          |
834
835         |'),nl,
836         write('          |
837
838         |'),nl,
839         write('          |
840
841         |'),nl,
842         write('          |
843
844         |'),nl,
845         write('          |
846
847         |'),nl,
848         write('          |
849
850         |'),nl,
851         write('          |
852
853         |'),nl,
854         write('          |
855
856         |'),nl,
857         write('          |
858
859         |'),nl,
860         write('          |
861
862         |'),nl,
863         write('          |
864
865         |'),nl,
866         write('          |
867
868         |'),nl,
869         write('          |
870
871         |'),nl,
872         write('          |
873
874         |'),nl,
875         write('          |
876
877         |'),nl,
878         write('          |
879
880         |'),nl,
881         write('          |
882
883         |'),nl,
884         write('          |
885
886         |'),nl,
887         write('          |
888
889         |'),nl,
890         write('          |
891
892         |'),nl,
893         write('          |
894
895         |'),nl,
896         write('          |
897
898         |'),nl,
899         write('          |
900
901         |'),nl,
902         write('          |
903
904         |'),nl,
905         write('          |
906
907         |'),nl,
908         write('          |
909
910         |'),nl,
911         write('          |
912
913         |'),nl,
914         write('          |
915
916         |'),nl,
917         write('          |
918
919         |'),nl,
920         write('          |
921
922         |'),nl,
923         write('          |
924
925         |'),nl,
926         write('          |
927
928         |'),nl,
929         write('          |
930
931         |'),nl,
932         write('          |
933
934         |'),nl,
935         write('          |
936
937         |'),nl,
938         write('          |
939
940         |'),nl,
941         write('          |
942
943         |'),nl,
944         write('          |
945
946         |'),nl,
947         write('          |
948
949         |'),nl,
950         write('          |
951
952         |'),nl,
953         write('          |
954
955         |'),nl,
956         write('          |
957
958         |'),nl,
959         write('          |
960
961         |'),nl,
962         write('          |
963
964         |'),nl,
965         write('          |
966
967         |'),nl,
968         write('          |
969
970         |'),nl,
971         write('          |
972
973         |'),nl,
974         write('          |
975
976         |'),nl,
977         write('          |
978
979         |'),nl,
980         write('          |
981
982         |'),nl,
983         write('          |
984
985         |'),nl,
986         write('          |
987
988         |'),nl,
989         write('          |
990
991         |'),nl,
992         write('          |
993
994         |'),nl,
995         write('          |
996
997         |'),nl,
998         write('          |
999
1000         |'),nl,
1001         write('          |
1002
1003         |'),nl,
1004         write('          |
1005
1006         |'),nl,
1007         write('          |
1008
1009         |'),nl,
1010         write('          |
1011
1012         |'),nl,
1013         write('          |
1014
1015         |'),nl,
1016         write('          |
1017
1018         |'),nl,
1019         write('          |
1020
1021         |'),nl,
1022         write('          |
1023
1024         |'),nl,
1025         write('          |
1026
1027         |'),nl,
1028         write('          |
1029
1030         |'),nl,
1031         write('          |
1032
1033         |'),nl,
1034         write('          |
1035
1036         |'),nl,
1037         write('          |
1038
1039         |'),nl,
1040         write('          |
1041
1042         |'),nl,
1043         write('          |
1044
1045         |'),nl,
1046         write('          |
1047
1048         |'),nl,
1049         write('          |
1050
1051         |'),nl,
1052         write('          |
1053
1054         |'),nl,
1055         write('          |
1056
1057         |'),nl,
1058         write('          |
1059
1060         |'),nl,
1061         write('          |
1062
1063         |'),nl,
1064         write('          |
1065
1066         |'),nl,
1067         write('          |
1068
1069         |'),nl,
1070         write('          |
1071
1072         |'),nl,
1073         write('          |
1074
1075         |'),nl,
1076         write('          |
1077
1078         |'),nl,
1079         write('          |
1080
1081         |'),nl,
1082         write('          |
1083
1084         |'),nl,
1085         write('          |
1086
1087         |'),nl,
1088         write('          |
1089
1090         |'),nl,
1091         write('          |
1092
1093         |'),nl,
1094         write('          |
1095
1096         |'),nl,
1097         write('          |
1098
1099         |'),nl,
1100         write('          |
1101
1102         |'),nl,
1103         write('          |
1104
1105         |'),nl,
1106         write('          |
1107
1108         |'),nl,
1109         write('          |
1110
1111         |'),nl,
1112         write('          |
1113
1114         |'),nl,
1115         write('          |
1116
1117         |'),nl,
1118         write('          |
1119
1120         |'),nl,
1121         write('          |
1122
1123         |'),nl,
1124         write('          |
1125
1126         |'),nl,
1127         write('          |
1128
1129         |'),nl,
1130         write('          |
1131
1132         |'),nl,
1133         write('          |
1134
1135         |'),nl,
1136         write('          |
1137
1138         |'),nl,
1139         write('          |
1140
1141         |'),nl,
1142         write('          |
1143
1144         |'),nl,
1145         write('          |
1146
1147         |'),nl,
1148         write('          |
1149
1150         |'),nl,
1151         write('          |
1152
1153         |'),nl,
1154         write('          |
1155
1156         |'),nl,
1157         write('          |
1158
1159         |'),nl,
1160         write('          |
1161
1162         |'),nl,
1163         write('          |
1164
1165         |'),nl,
1166         write('          |
1167
1168         |'),nl,
1169         write('          |
1170
1171         |'),nl,
1172         write('          |
1173
1174         |'),nl,
1175         write('          |
1176
1177         |'),nl,
1178         write('          |
1179
1180         |'),nl,
1181         write('          |
1182
1183         |'),nl,
1184         write('          |
1185
1186         |'),nl,
1187         write('          |
1188
1189         |'),nl,
1190         write('          |
1191
1192         |'),nl,
1193         write('          |
1194
1195         |'),nl,
1196         write('          |
1197
1198         |'),nl,
1199         write('          |
1200
1201         |'),nl,
1202         write('          |
1203
1204         |'),nl,
1205         write('          |
1206
1207         |'),nl,
1208         write('          |
1209
1210         |'),nl,
1211         write('          |
1212
1213         |'),nl,
1214         write('          |
1215
1216         |'),nl,
1217         write('          |
1218
1219         |'),nl,
1220         write('          |
1221
1222         |'),nl,
1223         write('          |
1224
1225         |'),nl,
1226         write('          |
1227
1228         |'),nl,
1229         write('          |
1230
1231         |'),nl,
1232         write('          |
1233
1234         |'),nl,
1235         write('          |
1236
1237         |'),nl,
1238         write('          |
1239
1240         |'),nl,
1241         write('          |
1242
1243         |'),nl,
1244         write('          |
1245
1246         |'),nl,
1247         write('          |
1248
1249         |'),nl,
1250         write('          |
1251
1252         |'),nl,
1253         write('          |
1254
1255         |'),nl,
1256         write('          |
1257
1258         |'),nl,
1259         write('          |
1260
1261         |'),nl,
1262         write('          |
1263
1264         |'),nl,
1265         write('          |
1266
1267         |'),nl,
1268         write('          |
1269
1270         |'),nl,
1271         write('          |
1272
1273         |'),nl,
1274         write('          |
1275
1276         |'),nl,
1277         write('          |
1278
1279         |'),nl,
1280         write('          |
1281
1282         |'),nl,
1283         write('          |
1284
1285         |'),nl,
1286         write('          |
1287
1288         |'),nl,
1289         write('          |
1290
1291         |'),nl,
1292         write('          |
1293
1294         |'),nl,
1295         write('          |
1296
1297         |'),nl,
1298         write('          |
1299
1300         |'),nl,
1301         write('          |
1302
1303         |'),nl,
1304         write('          |
1305
1306         |'),nl,
1307         write('          |
1308
1309         |'),nl,
1310         write('          |
1311
1312         |'),nl,
1313         write('          |
1314
1315         |'),nl,
1316         write('          |
1317
1318         |'),nl,
1319         write('          |
1320
1321         |'),nl,
1322         write('          |
1323
1324         |'),nl,
1325         write('          |
1326
1327         |'),nl,
1328         write('          |
1329
1330         |'),nl,
1331         write('          |
1332
1333         |'),nl,
1334         write('          |
1335
1336         |'),nl,
1337         write('          |
1338
1339         |'),nl,
1340         write('          |
1341
1342         |'),nl,
1343         write('          |
1344
1345         |'),nl,
1346         write('          |
1347
1348         |'),nl,
1349         write('          |
1350
1351         |'),nl,
1352         write('          |
1353
1354         |'),nl,
1355         write('          |
1356
1357         |'),nl,
1358         write('          |
1359
1360         |'),nl,
1361         write('          |
1362
1363         |'),nl,
1364         write('          |
1365
1366         |'),nl,
1367         write('          |
1368
1369         |'),nl,
1370         write('          |
1371
1372         |'),nl,
1373         write('          |
1374
1375         |'),nl,
1376         write('          |
1377
1378         |'),nl,
1379         write('          |
1380
1381         |'),nl,
1382         write('          |
1383
1384         |'),nl,
1385         write('          |
1386
1387         |'),nl,
1388         write('          |
1389
1390         |'),nl,
1391         write('          |
1392
1393         |'),nl,
1394         write('          |
1395
1396         |'),nl,
1397         write('          |
1398
1399         |'),nl,
1400         write('          |
1401
1402         |'),nl,
1403         write('          |
1404
1405         |'),nl,
1406         write('          |
1407
1408         |'),nl,
1409         write('          |
1410
1411         |'),nl,
1412         write('          |
1413
1414         |'),nl,
1415         write('          |
1416
1417         |'),nl,
1418         write('          |
1419
1420         |'),nl,
1421         write('          |
1422
1423         |'),nl,
1424         write('          |
1425
1426         |'),nl,
1427         write('          |
1428
1429         |'),nl,
1430         write('          |
1431
1432         |'),nl,
1433         write('          |
1434
1435         |'),nl,
1436         write('          |
1437
1438         |'),nl,
1439         write('          |
1440
1441         |'),nl,
1442         write('          |
1443
1444         |'),nl,
1445         write('          |
1446
1447         |'),nl,
1448         write('          |
1449
1450         |'),nl,
1451         write('          |
1452
1453         |'),nl,
1454         write('          |
1455
1456         |'),nl,
1457         write('          |
1458
1459         |'),nl,
1460         write('          |
1461
1462         |'),nl,
1463         write('          |
1464
1465         |'),nl,
1466         write('          |
1467
1468         |'),nl,
1469         write('          |
1470
1471         |'),nl,
1472         write('          |
1473
1474         |'),nl,
1475         write('          |
1476
1477         |'),nl,
1478         write('          |
1479
1480         |'),nl,
1481         write('          |
1482
1483         |'),nl,
1484         write('          |
1485
1486         |'),nl,
1487         write('          |
1488
1489         |'),nl,
1490         write('          |
1491
1492         |'),nl,
1493         write('          |
1494
1495         |'),nl,
1496         write('          |
1497
1498         |'),nl,
1499         write('          |
1500
1501         |'),nl,
1502         write('          |
1503
1504         |'),nl,
1505         write('          |
1506
1507         |'),nl,
1508         write('          |
1509
1510         |'),nl,
1511         write('          |
1512
1513         |'),nl,
1514         write('          |
1515
1516         |'),nl,
1517         write('          |
1518
1519         |'),nl,
1520         write('          |
1521
1522         |'),nl,
1523         write('          |
1524
1525         |'),nl,
1526         write('          |
1527
1528         |'),nl,
1529         write('          |
1530
1531         |'),nl,
1532         write('          |
1533
1534         |'),nl,
1535         write('          |
1536
1537         |'),nl,
1538         write('          |
1539
1540         |'),nl,
1541         write('          |
1542
1543         |'),nl,
1544         write('          |
1545
1546         |'),nl,
1547         write('          |
1548
1549         |'),nl,
1550         write('          |
15
```

```

11 write('      | | |      |__| | | | | | | ____| | | |
    | | | __ | | | ____| | | | | | | | | | | | | ') ,nl
12 write('      | | | ____ | | | | | | | | | | | | | ____|
    | | | |__| | | | ____ | | ____ | | ____ | | ____| | ') ,nl
13 write('      | | ____| | _| | _| | _| | _| | _| |
    ____| | | ____| | ____| | ____| | ____| | ____| |
    '),nl,
14 write('      |
        |'),nl,
15 write('      |
        |'),nl,
16 write('      |                               Francisca Leao |')
    Cerquinho Ribeiro da Fonseca ,nl,
17 write('      |
        Mariana Lopes Silva |'),nl,
18 write('      |
        |'),nl,
19 write('      |
        |'),nl,
20 write('      |                               1.Start |')
    Game Player vs Player ,nl,
21 write('      |
        |'),nl,
22 write('      |                               2.Start |'),
    Game PC vs Player nl,
23 write('      |
        |'),nl,
24 write('      |                               3.Start |'),nl
    Game PC vs PC ,
25 write('      |
        |'),nl,
26 write('      |                               4. |'),
    Set Difficulty nl,
27 write('      |
        |'),nl,
28 write('      |                               5. |'),
    How to Play nl,
29 write('      |

```

```

        |'),nl,
30 write('          |
                                6.Exit
                                |'),nl,
31 write('          |
        |'),nl,
32 write('          |
        |'),nl,
33 write('          |
        |-----|
        | '),nl,nl,nl,nl,
34 write('          Choose an
        option, please!
        nl, nl, nl, nl.
35
36
37 printHowToPlayMenu:-
38 nl,nl,nl,
39 write('
        |'),nl,
40 write('          |
        |'),nl,
41 write('          |
        |'),nl,
42 write('          |
        |'),nl,
43 write('          |
        |          - - -
        |          |'),nl,
44 write('          |          | | | |
        |          | | | | |
        |          |'),nl,
45 write('          |          | | | | |
        | | _ _ _ | | / / | _ _ _ _ _ |
        |),nl,
46 write('          |          | _ | / _ | / / /
        | _ / _ | | _ / | / _ ' | | | |
        |),nl,
47 write('          |          | | | | ( _ ) | V V /
        | | | ( _ ) | | | | ( _ ) | | |
        |),nl,
48 write('          |          | _ | | _ / | _ / | _ /
        | _ | _ _ / | | | _ | _ _ , | | _ _ , |
        |'),nl,
49 write('          |
        _ _ / |          |'),nl,
50 write('          |

```

```

51         |___/                                     |'),nl,
write('          |

        |'),nl,
52 write('          |      In this game, each player tries to
        remove as much as possible from their of the board. |
        '),nl,
53 write('          |In turn, the player must jump with his
        piece to another and if the piece that jumped is one|
        '),nl,
54 write('          |of yours, you should remove it from the
        game. Otherwise, if it is one of the adversaries, |'
        ),nl,
55 write('          |then you can remove any of your pieces
        ohe board (including the one used in the jump).      |
        '),nl,
56 write('          |      Each player can "chain" up to 3
        jumps with the same piece of but you can not jump
        over |'),nl,
57 write('          |it twice. The piece with which the
        player jumps can not occupy the same space during the
        |'),nl,
58 write('          |same move. If not can jump, the player
        can ignore his turn. The game ends when a player has|
        '),nl,
59 write('          |no more pieces on the board or no player
        can make a real leap.                                |'
        ),nl,
60 write('          |      At the end of the game, each player
        gets 1 point for each of their pieces out of your |'
        ),nl,
61 write('          |starting area and 3 points for each
        piece that is in your area of match. The player with
        |'),nl,
62 write('          |the lowest number of points wins.

        |'),nl,
63 write('          |

        |'),nl,
64 write('          |

        |'),nl,
65 write('          |

                                0. Back
                                |'),nl,
66 write('          |

        -----
        | '),nl,nl,nl,nl.

67
68 printSetLevelMenu:-
69     nl,nl,nl,
70     write('

        -----
71     write('          |

```

```

72         |'),nl,
write('          |

73         |'),nl,
write('          |          Please
        choose the level of the Game:
              |'),nl,
74 write('          |

75         |'),nl,
write('          |

        Difficulty          1. Normal
76 write('          |          |'),nl,

        Difficulty          2. Hard
77 write('          |          |'),nl,

78         |'),nl,
write('          |

79         |'),nl,
write('          |

        -----
        | '),nl,nl,nl,nl.

80
81 mainMenu :- printMainMenu,
82 now(X), setrand(X),
83 read(Input),
84 set_mode_game(Input),
85 readInput(Input).
86
87 readInput(0) :- mainMenu.
88
89 readInput(1) :- initialBoard(Board),printFinalBoard(
90     Board),
91 play(Board),
92 mainMenu.
93
94 readInput(2) :- initialBoard(Board), printFinalBoard(
95     Board),
96 set_user_is('pcX'),
97 play(Board),
98 mainMenu.
99
100 readInput(3) :- initialBoard(Board), printFinalBoard(
101     Board),
102 set_user_is('pcX'),
103 play(Board),
104 mainMenu.
105
106 readInput(4) :- printSetLevelMenu,
read(Input),
readInput2(Input).

```



```

107     readInput2(1) :- set_level(1),
108     mainMenu.
109
110     readInput2(2) :- set_level(2),
111     mainMenu.
112
113     readInput(5) :- printHowToPlayMenu,
114     read(Input),
115     readInput(Input).
116
117     readInput(6) :- write('Exiting...').

```

## Anexo VI

### Ficheiro "utilities.pl"

```

1  %Predicate that read a char
2  getChar(Input) :- get_char(_Input),
3                    get_char(Input).
4
5  %Predicate that read a number
6  getCode(Input) :- get_code(_TempInput),
7                    get_code(TempInput),
8                    Input is TempInput - 48.
9
10 :-dynamic player/1.
11 :-dynamic mode_game/1.
12 :-dynamic user_is/1.
13 :-dynamic level/1.
14
15 mode_game(1).
16 player(playerX).
17 user_is(player).
18 level(1).
19
20 %Predicate that sets the player
21 set_player(Player):-
22     nonvar(Player),
23     retract(player(_)),
24     asserta(player(Player)).
25
26 %Predicate that sets the mode game
27 set_mode_game(Newmode):-
28     nonvar(Newmode),
29     integer(Newmode),
30     retract(mode_game(_)),
31     asserta(mode_game(Newmode)).
32
33 %Predicate that sets the user
34 set_user_is(NewPlayer):-
35     nonvar(NewPlayer),
36     retract(user_is(_)),
37     asserta(user_is(NewPlayer)).
38
39 %Predicate that sets the level
40 set_level(Level):-
41     nonvar(Level),

```

```

42     integer(Level),
43     retract(level(_)),
44     asserta(level(Level)).
45
46 %Predicate that converts each letter into its respective
    number
47     letterToNumber('A',1).
48     letterToNumber('B',2).
49     letterToNumber('C',3).
50     letterToNumber('D',4).
51     letterToNumber('E',5).
52     letterToNumber('F',6).
53     letterToNumber('G',7).
54     letterToNumber('H',8).
55     letterToNumber('I',9).
56
57 %Predicate that converts each number into its respective
    letter
58     numberToLetter(1,'A').
59     numberToLetter(2,'B').
60     numberToLetter(3,'C').
61     numberToLetter(4,'D').
62     numberToLetter(5,'E').
63     numberToLetter(6,'F').
64     numberToLetter(7,'G').
65     numberToLetter(8,'H').
66     numberToLetter(9,'I').
67
68 %Predicate copying one board to another
69 duplicate(_Old,_New):-fail.
70 duplicate(_Old,_Old).

```

## Anexo VII

### Ficheiro "validateMoves.pl"

```

1 %Predicate that validates the plays of the pc in the area X1
2 validateMovePC('areaX1',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else(((LastCol==5,LastRow==3);(LastCol==5,LastRow
        ==4);(LastCol==5,LastRow==2)),
3 (Nrow is LastRow+2,
4 Ncol is LastCol),
5 ((Ncol is LastCol+2,
6 Nrow is LastRow+2,
7 getPiece(Board,Nrow,Ncol,Piece),
8 Piece\='empty'),
9 (Nrow is LastRow,
10 Ncol is LastCol+2,
11 getPiece(Board,Nrow,Ncol,Piece),
12 Piece\='empty'),
13 (Nrow is LastRow+2,
14 Ncol is LastCol,
15 getPiece(Board,Nrow,Ncol,Piece),
16 Piece\='empty'))),
17 if_then_else((LastCol==4,LastRow==3),
18 (Nrow is LastRow+2,
19 Ncol is LastCol+2),true).

```

```

20 %Predicate that validates the plays of the pc in the area X2
21 validateMovePC('areaX2',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else((LastCol==2,LastRow==5),
22     (Nrow is LastRow+2,
23     Ncol is LastCol;
24     (Nrow is LastRow,
25     Ncol is LastCol+2)),
26     ((Nrow is LastRow-2,
27     Ncol is LastCol+2,
28     getPiece(Board,Nrow,Ncol,Piece),
29     Piece\='empty');
30     (Nrow is LastRow,
31     Ncol is LastCol+2,
32     getPiece(Board,Nrow,Ncol,Piece),
33     Piece\='empty');
34     (Nrow is LastRow+2,
35     Ncol is LastCol,
36     getPiece(Board,Nrow,Ncol,Piece),
37     Piece\='empty'))),
38
39     if_then_else((LastCol==3,LastRow==5),
40     (Nrow is LastRow,
41     Ncol is LastCol+2),true),
42
43     if_then_else((LastCol==3,LastRow==6),
44     (Nrow is LastRow-2,
45     Ncol is LastCol+2),true).
46
47 %Predicate that validates the plays of the pc in the area Y1
48 validateMovePC('areaY1',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else(((LastCol==7,LastRow==5);(LastCol==8,LastRow
    ==5);(LastCol==6,LastRow==5)),
49     (Ncol is LastCol-2,
50     Nrow is LastRow),
51     ((Ncol is LastCol-2,
52     Nrow is LastRow+2,
53     getPiece(Board,Nrow,Ncol,Piece),
54     Piece\='empty');
55     (Ncol is LastCol,
56     Nrow is LastRow+2,
57     getPiece(Board,Nrow,Ncol,Piece),
58     Piece\='empty');
59     (Ncol is LastCol-2,
60     Nrow is LastRow,
61     getPiece(Board,Nrow,Ncol,Piece),
62     Piece\='empty'))),
63
64     if_then_else((LastCol==7,LastRow==4),
65     (Nrow is LastRow+2,
66     Ncol is LastCol-2),true).
67
68 %Predicate that validates the plays of the pc in the area Y2
69 validateMovePC('areaY2',LastCol,LastRow,Ncol,Nrow,Board) :-
70     if_then_else(((LastCol==5,LastRow==8);(LastCol==5,LastRow
    ==7);(LastCol==5,LastRow==6)),
71     (Ncol is LastCol,

```

```

72 Nrow is LastRow-2),
73 ((Ncol is LastCol-2,
74 Nrow is LastRow-2,
75 getPiece(Board,Nrow,Ncol,Piece),
76 Piece\='empty');
77 (Ncol is LastCol-2,
78 Nrow is LastRow,
79 getPiece(Board,Nrow,Ncol,Piece),
80 Piece\='empty');
81 (Ncol is LastCol,
82 Nrow is LastRow-2,
83 getPiece(Board,Nrow,Ncol,Piece),
84 Piece\='empty'))),
85
86 if_then_else((LastCol==6,LastRow==5),
87 (Ncol is LastCol-2,
88 Nrow is LastRow),true).
89 %Predicate that validates the plays of the player in the
    area X1
90 validateMove('areaX1',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else((LastCol==5,LastRow==3);(LastCol==5,
    LastRow==4);(LastCol==5,LastRow==2)),
91 (RowTemp is LastRow+2,
92 Nrow == RowTemp,
93 Ncol == LastCol),
94 ((ColTemp is LastCol+2,
95 RowTemp is LastRow+2,
96 Ncol == ColTemp,
97 Nrow == RowTemp,
98 getPiece(Board,Nrow,Ncol,Piece),
99 Piece\='empty');
100 (ColTemp is LastCol+2,
101 Nrow == LastRow,
102 Ncol == ColTemp,
103 getPiece(Board,Nrow,Ncol,Piece),
104 Piece\='empty');
105 (RowTemp is LastRow+2,
106 Nrow == RowTemp,
107 Ncol == LastCol,
108 getPiece(Board,Nrow,Ncol,Piece),
109 Piece\='empty'))),
110 if_then_else((LastCol==4,LastRow==3),
111 (RowTemp is LastRow+2,
112 ColTemp is LastCol+2,
113 Nrow == RowTemp,
114 Ncol == ColTemp),true).
115
116 %Predicate that validates the plays of the player in the
    area X2
117 validateMove('areaX2',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else((LastCol==2,LastRow==5),
118 (RowTemp is LastRow+2,
119 Nrow == RowTemp,
120 Ncol == LastCol;
121 (ColTemp is LastCol+2,
122 Nrow == LastRow,

```

```

123     Ncol == ColTemp)),
124     ((ColTemp is LastCol+2,
125     RowTemp is LastRow-2,
126     Nrow == RowTemp,
127     Ncol == ColTemp,
128     getPiece(Board,Nrow,Ncol,Piece),
129     Piece\='empty'));
130     (ColTemp is LastCol+2,
131     Nrow == LastRow,
132     Ncol == ColTemp,
133     getPiece(Board,Nrow,Ncol,Piece),
134     Piece\='empty'));
135     (RowTemp is LastRow+2,
136     Nrow == RowTemp,
137     Ncol == LastCol,
138     getPiece(Board,Nrow,Ncol,Piece),
139     Piece\='empty'))),
140
141     if_then_else((LastCol==3,LastRow==5),
142     (ColTemp is LastCol+2,
143     Nrow == LastRow,
144     Ncol == ColTemp),true),
145
146     if_then_else((LastCol==3,LastRow==6),
147     (ColTemp is LastCol+2,
148     RowTemp is LastRow-2,
149     Nrow == RowTemp,
150     Ncol == ColTemp),true).
151
152     %Predicate that validates the plays of the player in the
153     area Y1
154     validateMove('areaY1',LastCol,LastRow,Ncol,Nrow,Board) :-
155         if_then_else(((LastCol==7,LastRow==5);(LastCol==8,
156         LastRow==5);(LastCol==6,LastRow==5)),
157         (ColTemp is LastCol-2,
158         Ncol == ColTemp,
159         Nrow == LastRow),
160         ((ColTemp is LastCol-2,
161         Ncol == ColTemp,
162         RowTemp is LastRow+2,
163         Nrow == RowTemp,
164         getPiece(Board,Nrow,Ncol,Piece),
165         Piece\='empty');
166         (Ncol == LastCol,
167         RowTemp is LastRow+2,
168         Nrow == RowTemp,
169         getPiece(Board,Nrow,Ncol,Piece),
170         Piece\='empty');
171         (ColTemp is LastCol-2,
172         Ncol == ColTemp,
173         Nrow == LastRow,
174         getPiece(Board,Nrow,Ncol,Piece),
175         Piece\='empty'))),

```

```

176 Nrow == RowTemp,
177 ColTemp is LastCol-2,
178 Ncol == ColTemp),true).
179
180 %Predicate that validates the plays of the player in the
    area Y2
181 validateMove('areaY2',LastCol,LastRow,Ncol,Nrow,Board) :-
    if_then_else(((LastCol==5,LastRow==8);(LastCol==5,
        LastRow==7);(LastCol==5,LastRow==6)),
182 (RowTemp is LastRow-2,
183 Ncol == LastCol,
184 Nrow == RowTemp),
185 ((ColTemp is LastCol-2,
186 RowTemp is LastRow-2,
187 Ncol == ColTemp,
188 Nrow == RowTemp,
189 getPiece(Board,Nrow,Ncol,Piece),
190 Piece\='empty');
191 (ColTemp is LastCol-2,
192 Ncol == ColTemp,
193 Nrow == LastRow,
194 getPiece(Board,Nrow,Ncol,Piece),
195 Piece\='empty');
196 (RowTemp is LastRow-2,
197 Ncol == LastCol,
198 Nrow == RowTemp,
199 getPiece(Board,Nrow,Ncol,Piece),
200 Piece\='empty'))),
201
202 if_then_else((LastCol==6,LastRow==5),
203 (ColTemp is LastCol-2,
204 Ncol == ColTemp,
205 Nrow == LastRow),true).

```