Universidade do Porto Faculdade de Engenharia FEUP	Faculdade de Engenharia da Universidade do Porto **Mestrado Integrado em Eng. Informática e Computação** **Programação em Lógica** *Época Normal – Com Consulta / Duração: 2h30m*		**2007/2008** **3º Ano MIEIC**
Nome:		**Data:**	**15/01/2008**

Nota: Não utilize nenhuma biblioteca do Prolog, excepto a CLPFD, para resolver o exame. Predicados definidos em alíneas anteriores (mesmo que não os implemente) podem ser utilizados em alíneas seguintes.

GRUPO I – Participações em Filmes (4 Val)

1) Suponha que o seguinte conjunto de factos é ilustrativo de uma base de conhecimento sobre filmes e actores, onde é indicado o nome da pessoa, o nome do filme/série em que participou, o seu papel, e em caso de ser actor, o nome do personagem interpretado:

```
in(terryOQuinn, lost, actor, johnLocke).
in(danielDaeKim, lost, actor, jinKwon).
in(jjAbrams, lost, creator).
in(zacharyQuinto, heroes, actor, sylar).
in(davidAnders, alias, actor, julianSark).
in(gregGrunberg, lost, guestStar, pilot).
in(gregGrunberg, alias, actor, ericWeiss).
in(terryOQuinn, alias, guestStar, directorKendall).
in(danielDaeKim, 24, guestStar, tomBaker).
in(jjAbrams, alias, creator).
in(zacharyQuinto, 24, actor, adamKaufman).
in(davidAnders, heroes, guestStar, adamMonroe).
in(gregGrunberg, heroes, actor, mattParkman).
```

1.1) Construa o predicado ***cast(+Name, -List),*** que devolve em List uma lista com todos os participantes num determinado filme/série. A lista deve conter três sublistas as quais contêm os criadores ("creator"), os actores ("actor"), e os actores convidados ("guestStar"), respectivamente. O predicado deve ainda imprimir o elenco de forma adequada, conforme o exemplo abaixo. **(1,5 Val)**

Exemplo:

```
?- cast(lost, List).
Creators:
      jjAbrams
Actors:
      terryOQuinn as johnLocke
      danielDaeKim as jinKwon
Guest Stars:
      gregGrunberg as pilot

List = [ [jjAbrams], [terryOQuinn, danielDaeKim], [gregGrunberg] ]
```

1.2) Construa o predicado ***workedTogether(+Name1, +Name2),*** que sucede caso as duas pessoas indicadas nos argumentos tenham trabalhado juntas num mesmo filme/série. **(1 Val)**

Exemplo:

```
?- workedTogether(terryOQuinn, zacharyQuinto).
No.
?- workedTogether(terryOQuinn, davidAnders).
Yes.
```

1.3) Construa o predicado ***degrees(+Name1, +Name2, -Degrees, -Contacts, -Where),*** que devolve em ***Degrees*** o número de graus de separação entre duas pessoas, em ***Contacts*** as pessoas que os separam, em ***Where*** os filmes/séries em que trabalharam juntos, pela mesma ordem (de notar que esta lista tem sempre mais um elemento que a de contactos. Considera-se que se duas pessoas trabalharam juntas, o seu grau de separação é 0. *Nota: Não é necessário garantir que o grau de separação seja o mínimo, mas sim um possível.* **(1,5 Val)**

Exemplo:

```
?- degrees(terryOQuinn, davidAnders, Degrees, Contacts, Where).
Degrees=0, Contacts=[], Where=[alias]    %(Trabalharam juntos em Alias)
?- degrees(terryOQuinn, zacharyQuinto, Degrees, Contacts, Where).
```

```
Degrees=1, Contacts=[danielDaeKim], Where=[lost,24]   %(O'Quinn
trabalhou com Daniel Dae Kim em Lost, o qual trabalhou com Zachary Quinto em
24)
```

GRUPO II – Programação em Prolog (4,5 Val)

2.1) Construa o predicado ***tree(+N)***, que desenha uma árvore de natal, com o tamanho indicado. *Nota: Tenha em atenção que os asteriscos são separados por espaços no interior da árvore* **(1,5 Val).**

Exemplo:

```
| ?- tree(1).
*
*

| ?- tree(2).
 *
* *
 *

| ?- tree(3).
  *
 * *
* * *
  *

| ?- tree(5).
    *
   * *
  * * *
 * * * *
* * * * *
    *

| ?- tree(6).
     *
    * *
   * * *
  * * * *
 * * * * *
* * * * * *
     *
     N
     N
```

2.2) Segundo as regras de admissão ao novo clube VIP da cidade do Porto:

- Um candidato tem de ter uma idade entre 18 e 30 anos (inclusive) e pelo menos um filho/a;
- Um candidato tem de ter dois proponentes, ambos membros do clube há pelo menos dois anos;
- Nenhum dos proponentes pode ser progenitor do candidato.

Defina uma Base de Factos simples para guardar a informação relativa aos membros clube (que contenha já, para todos os membros do clube, os respectivos filhos e outros dados que considere necessários) e escreva um predicado PROLOG ***aceitavel(Nome, Idade, Prop1, Prop2, ListaFilhos)*** que lhe permita decidir se um candidato é aceitável para membro do clube. Suponha que em caso de aceitação do candidato, o predicado deve suceder e a base de factos deve ser actualizada em consonância (adição do novo membro e respectivos dados). Em caso de rejeição, o predicado ***aceitavel/5*** deve falhar. **(1.5 Val)**

2.3) Considere o seguinte ficheiro Prolog: **(1.5 Val)**

a) Indique todas as respostas, por ordem, que o objectivo ***?- p(X)***. produz.

b) Se substituirmos a linha 2 (***p(X):- b(X), c(X), d(X).***) por ***p(X):- b(X), c(X), !, d(X).*** que respostas obtém agora o objectivo ***?- p(X)***. ?

c) Apresente as respectivas árvores de pesquisa justificando a diferença. Justifique também se o Cut introduzido é Verde ou Vermelho.

```
p(X):- a(X).                    %1
p(X):- b(X), c(X), d(X).        %2
p(X):- e(X).                    %3
a(1).
b(1).
c(1).
b(2).
c(2).
d(2).
e(3).
```

GRUPO III – Puzzle Infantil (5,5 Val)

Imagine como exemplo que tem o típico puzzle infantil (conhecido como N-Puzzle) de três por três, em que as peças podem deslizar para a adjacente quando vazia, com 8 casas que formam uma imagem, inicialmente fora de ordem, representado como uma lista de listas:

Puzzle


```
[ [3, 6, 8],
  [5, 0, 7],
  [2, 1, 4] ]
```

Solução


```
[ [1, 2, 3],
  [4, 5, 6],
  [7, 8, 0] ]
```

3.1) Considerando que o predicado ***get(-Elem, +List, +X, +Y)***, que devolve o elemento que se encontra na posição (X, Y) na lista de listas *List* já se encontra implementado, defina o predicado ***set(+Elem, +List, +X, +Y, -NewList)***, que altera a lista de listas *List* de forma a que a posição (X,Y) passe a ser ocupada por *Elem*, devolvendo a lista alterada em *NewList*. **(1 Val)**

Exemplo:

```
?- set(2, [[3,0,8],[5,6,7],[2,1,4]], 2, 3, NL).
NL = [[3,0,8],[5,6,7],[2,2,4]]
?- set(0, [[3,0,8],[5,6,7],[2,1,4]], 2, 2, NL).
NL = [[3,0,8],[5,0,7],[2,1,4]]
```

3.2) Construa o predicado ***exchange(+List, +Xi, +Yi, +Xf, +Yf, -NewList),*** que troca as posições dos elementos em (Xi, Yi) e (Xf, Yf) na lista de listas *List*, devolvendo a lista alterada em *NewList*. Deve ser verificado se (Xi, Yi) e (Xf, Yf) são posições adjacentes, se ambas estão dentro do tabuleiro, e ainda se uma das casas é vazia. **(1 Val)**

Exemplo:

```
?- exchange( [[3,6,8],[5,0,7],[2,1,4]], 2, 1, 2, 2, NewList).
NewList = [[3,0,8],[5,6,7],[2,1,4]]
?- exchange( [[3,6,8],[5,0,7],[2,1,4]], 2, 1, 3, 3, NewList).
no.
```

3.3) Construa o predicado ***find(+Elem, +List, -X, -Y),*** que encontra o primeiro elemento igual a *Elem* numa lista de listas *List*, devolvendo as coordenadas *X* e *Y* do mesmo. **(1 Val)**

Exemplo:

```
?- find(0, [[3,6,8],[5,0,7],[2,1,4]], X, Y).
X = 2, Y = 2
?- find(0, [[3,0,8],[5,6,7],[2,1,4]], X, Y).
X = 2, Y = 1
```

3.4) Construa o predicado ***operator(+Initial, -Final),*** que dada uma determinada configuração inicial, devolva uma a uma, para todas as possibilidades de movimentações válidas os respectivos tabuleiros resultantes. **(1 Val)**

Exemplo:

```
?- operator([[3,6,8],[5,0,7],[2,1,4]], Final).
Final = [[3,0,8],[5,6,7],[2,1,4]] ;
Final = [[3,6,8],[0,5,7],[2,1,4]] ;
Final = [[3,6,8],[5,7,0],[2,1,4]] ;
Final = [[3,6,8],[5,1,7],[2,0,4]] ;
no.
```

3.5) Construa o predicado ***solve(+List, +FinalList, -Operat),*** que devolve em *Operat* o conjunto de estados por onde passa o puzzle para ser resolvido, evitando estados repetidos. **(1.5 Val)**

Exemplo:

```
?- solve([[1,2,3],[4,5,6],[0,7,8]], [[1,2,3],[4,5,6],[7,8,0]],
Operat).
Operat = [ [[1,2,3],[4,5,6],[7,0,8]] ]
?- solve([[1,2,3],[0,5,6],[4,7,8]], [[1,2,3],[4,5,6],[7,8,0]],
Operat).
Operat = [[[1,2,3],[4,5,6],[0,7,8]], [[1,2,3],[4,5,6],[7,0,8]] ]
```

GRUPO IV – Programação em Lógica com Restrições (6 val)

Nota: Não esquecer de utilizar a biblioteca clpfd e que as variáveis utilizadas são inteiras (domínio finito).

4.1) O professor de lógica matemática do Joãozinho colocou um problema na aula. Os alunos tinham de descobrir quais os 6 números que estavam de acordo com as seguintes restrições:

- Os números são inteiros positivos menores que 50, ordenados de forma crescente.
- O primeiro número é igual a metade do segundo.
- O quarto número é o dobro do segundo.
- A soma de todos os números excepto o segundo é igual a 100.
- O quinto número é igual à soma do segundo e terceiro números.
- Dois dos seis números são consecutivos.
- A soma do primeiro e terceiro números é igual à diferença entre o último e penúltimo números.

De forma a ser mais rápido que os colegas, o Joãozinho utilizou PLR para descobrir estes 6 números. Use também PLR para os descobrir. **(1,5 Val)**

4.2) Suponha a seguinte multiplicação em que os Is representam algarismos ímpares e os Ps representam algarismos pares (>0). Utilizando Programação em Lógica com Restrições, resolva o problema, determinando o valor de todos os Is e Ps. *Sugestão: Utilize o predicado list_to_fd_set para construir os domínios das variáveis.* (**1,5 Val**)

	I_1	P_1	P_2
	x	P_3	P_4
P_5	I_2	P_6	P_7
P_8	I_3	P_9	
I_4	I_5	P_{10}	P_{11}

4.3) Suponha uma empresa que pretende definir a sua produção de um dado conjunto de produtos nas suas diversas fábricas para o próximo mês. A empresa dispõe de um dado conjunto de fábricas e cada fábrica possui restrições de capacidade (mínima e máxima) de produção de determinados produtos. Encontram-se também disponíveis dados relativos ao custo de produzir uma unidade de cada produto em cada fábrica (0 em caso de não ser possível a produção). Suponha que os dados são apresentados no formato seguinte:

```
fabricas(3).          // dispomos de 3 fábricas
produtos(6).          // produção de 6 produtos
fabrica(1,[2-(20,100), 3-20, 4-(20,30), 5-50]). // fábrica 1 não pode produzir produto 1, pode produzir 20 a 100 unidades do produto 2, produz 20 do produto 3, porduz 20 a 30 do 4, 50 do 5 e não pode produzir o 6.
fabrica(2,[1-(0,100), 3-(0,20), 4-(10,20), 5-(20,50), 6-(10,50)]).
fabrica(3,[1-(0,100), 2-(50,100), 3-(0,20), 4-(10,20), 5-(20,50), 6-(10,50)]).

produtos(1,[0,5,3],10,(20,150)).  //produto 1 custa 5 a ser produzido na fabrica 2 e custa 3 a ser produzido na fábrica 3. Cada unidade vende-se por 10, a quantidade minima a produzir é 20 e a máxima é 150.
produtos(2,[7,4,5],13,(10,100)).
produtos(3,[2,2,1],5,(60,250)).
produtos(4,[5,0,3],8,(30,80)).
produtos(5,[1,3,2],5,(80,350)).
produtos(6,[10,12,0],20,(10,80)).
```

Pretende-se construir um programa utilizando PLR que permita determinar o escalonamento de produção mensal, ou seja, quanto vai cada fábrica produzir de cada produto, no próximo mês, maximizando o respectivo lucro global (preço de venda menos os custos de produção) **(3 Val).**

BOM TRABALHO!