Universidade do Porto Faculdade de Engenharia **FEUP**	Faculdade de Engenharia da Universidade do Porto **Mestrado Integrado em Eng. Informática e Computação** **Mestrado Integrado Eng. Electrotécnica e de Computadores** **Programação em Lógica** *Época de Recurso – Com Consulta / Duração: 2h30m*	**2011/2012** **3º Ano MIEIC** **4º/5º Ano MIEEC** **10/02/2012**

Notas:

a) No Grupo I use apenas Prolog Standard, e no Grupo II use também a biblioteca CLPFD do SICStus Prolog.

b) Predicados solicitados em alíneas anteriores podem ser utilizados em alíneas seguintes, mesmo que não os tenha implementado.

c) Responda a cada questão (agrupamento de alíneas) numa folha separada: N questões = N folhas.

GRUPO I – Programação em Prolog (13 valores)

1. [5 valores]
Considere a seguinte base de dados com informação sobre objetos dispostos numa sala. O predicado ***made_of(Obj, Mat)*** indica que o objeto ***Obj*** é feito de ***Mat***. O predicado ***on(Obj1, Obj2)*** indica que ***Obj1*** está diretamente em cima de ***Obj2***. Por omissão, os objetos estão colocados no chão.

```
object(ball).      | made_of(ball, plastic).    | on(tv, table).
object(sofa).      | made_of(sofa, tissue).     | on(ball, table).
object(table).     | made_of(table, wood).      | on(pillow, sofa).
object(statue).    | made_of(statue, wood).     | on(book, pillow).
object(tv).        | made_of(tv, plastic).      | on(statue, tv).
object(pillow).    | made_of(pillow, tissue).   |
object(book).      | made_of(book, paper).      |
```

Em cada um dos predicados que se pedem de seguida, construa sempre soluções genéricas, isto é, que funcionem para quaisquer objetos presentes no espaço e em qualquer configuração.

a) Implemente o predicado ***under(?X, ?Y)***, que sucede se o objeto ***X*** está diretamente debaixo do objeto ***Y***. Por exemplo, ***sofa*** está diretamente debaixo de ***pillow***.

b) Implemente o predicado ***below(?X, ?Y)***, que sucede se o objeto ***X*** está por baixo do objeto ***Y***. Por exemplo, ***table*** está por baixo de ***statue***.

c) Implemente o predicado ***unique(?X)***, que sucede se o objeto ***X*** for o único feito no seu material.

d) Implemente o predicado ***on_same(?X, ?Y, ?Z)***, que sucede se ***X*** e ***Y*** estão diretamente em cima do mesmo objeto ***Z***. Se estiverem ambos no chão, ***Z*** deve ser unificado com o átomo ***floor***.

e) Implemente o predicado ***describe***, sem argumentos, que imprime no ecrã, por *backtracking*, frases do tipo "the X is made of Y", para todos os objetos existentes. No final o predicado deve suceder.

f) Implemente o predicado ***down_to_floor(?X, ?L)***, que unifica L com a lista de objetos, por ordem, que separam o objeto X do chão. Exemplo:

```
?- down_to_floor(statue, L).
L = [statue,tv,table]
```

g) Pretende-se saber se a tv está sobre um objeto de madeira. Para isso, alguém fez a pergunta ***?- made_of(X, madeira), on(tv, X).*** Comente sobre o esforço de pesquisa desta pergunta e sugira uma formulação alternativa.

2. [5 valores]

Suponha que tem uma base de conhecimento em Prolog com dois tipos de factos:

- ***series(Series, Network, YearFirst)***: uma série televisiva (***Series***), o canal onde passa (***Network***) e o ano do seu primeiro episódio (***YearFirst***).
- ***episode(Episode, Series, MillionViews)***: um episódio ***Episode*** de uma série ***Series*** teve já ***MillionViews*** milhões de visualizações.

Exemplo:

```
series('Fringe','FOX',2008).
series('Game of Thrones','HBO',2011).
series('How I Met Your Mother','CBS',2005).
series('House M.D.','FOX',2004).
series('Prison Break','FOX',2005).
series('True Blood','HBO',2008).
series('CSI','CBS',2000).
```

```
episode('Peter','Fringe',5.97).
episode('The Same Old Story','Fringe',13.27).
episode('Enemy of My Enemy','Fringe',3.19).
episode('Winter is Coming','Game of Thrones',2.22).
episode('Fire and Blood','Game of Thrones',3.04).
episode('Allen','Prison Break',10.51).
episode('Tweener','Prison Break',9.01).
episode('You Smell Like Dinner','True Blood',2.90).
```

a) Implemente o predicado ***filterSeries(+ListOfSeries, +Year, -List)***, que devolve, em ***List***, uma lista com as séries da lista ***ListOfSeries*** que obedecem a um destes critérios:

- começou após ***Year*** e tem pelo menos dois episódios;
- começou antes de ***Year*** e tem pelo menos um episódio com mais de 4 milhões de visualizações;
- começou no ano ***Year***.

Exemplos:

```
?- filterSeries(['Fringe','How I Met Your Mother','House M.D.'], 2004, L).
L = ['Fringe', 'House M.D.'].
?- filterSeries(['Game of Thrones','Prison Break','True Blood'], 2007, L).
L = ['Game of Thrones', 'Prison Break'].
```

b) Implemente o predicado ***mostAndLeastSeen(+ListOfSeries, -List)*** que, considerando as séries na lista ***ListOfSeries***, obtém uma lista com os episódios mais vistos, um para cada série, seguido dos episódios menos vistos, um para cada série. Cada elemento da lista deve ser da forma ***Series-Episode***. Em caso de empate nos episódios de uma série com mais (menos) visualizações, o mais (menos) visto é um qualquer dos empatados. Exemplo:

```
?- mostAndLeastSeen(['Fringe', 'True Blood'], L).
L = ['Fringe'-'The Same Old Story', 'True Bllod'-'You Smell Like Dinner',
    'Fringe'-'Enemy of My Enemy', 'True Blood'-'You Smell Like Dinner']
```

c) Explique o funcionamento do predicado ***p(+X, +Y, -List)*** apresentado em baixo. Indique o resultado da pergunta ***?- p(2, 11, List).*** usando a base de conhecimento ilustrada acima.

```
p(X, Y, List):-
      findall(A-B, (series(_, A, _),
                    findall(C, (series(C,A,_),
                                episode(D, C, D1), episode(E, C, E1),
                                D\=E, D1>X, D1<Y, E1>X, E1<Y), F),
                    setof(G, member(G, F), B)), H),
      setof(I, member(I, H), List).
```

3. [3 valores]

Pretende-se implementar uma calculadora que funcione apenas com números inteiros positivos. São requeridas apenas as seguintes seis operações: soma (+), subtração (-), multiplicação (*), divisão inteira (/), resto da divisão (//) e exponenciação (^).

Os números são representados como sucessores: 0 é representado como ***0***, 1 é representado como ***s(0)***, 2 é representado como ***s(s(0))***, e assim sucessivamente.

As operações são definidas através de notação prefixa, em que o operador matemático antecede as expressões a avaliar. Por exemplo, 1 + 2 = ***(+ 1 2)***, ou 3 + 4 * 2 = ***(* (+ 3 4) 2)***. Assim, nesta calculadora uma operação matemática é representada por um terno ***(op, exp_left, exp_right)***, onde ***op*** é o operador, e ***exp_left*** e ***exp_right*** são expressões, podendo ser um número inteiro ou qualquer operação válida.

a) Implemente o predicado ***evaluate(+Expression, -Result)***, que determina o valor da expressão ***Expression*** fornecida. Faça uso exclusivo da notação de sucessores para representação dos números. Por exemplo:

```
?- evaluate( (+, s(0), s(s(0))) , X).   % 1 + 2 = 3
X = s(s(s(0)))
?- evaluate( (*, s(s(0)) , (+, s(0), s(s(0)))) , X).   % 2 * (1 + 2) = 6
X = s(s(s(s(s(0)))))
```

b) Implemente o predicado ***translate(+Expression)***, que imprime no ecrã a expressão fornecida na notação prefixa e com notação de sucessores, usando notação infixa (a notação usada normalmente) e algarismos árabes. A ordem e precedência das operações deve ser respeitada, e os parêntesis devem ser reduzidos ao mínimo necessário. Pode assumir que os seguintes predicados estão já implementados:

- ***precede(+Op1, +Op2)***: sucede se o operador ***Op1*** tem maior precedência que o operador ***Op2***.
- ***successor2arab(+Successor, -Arab)*** - traduz um número da notação de sucessores para a notação árabe.

Exemplos:

```
?- translate((+,(//,(s(s(s(s(0)))),s(s(0)))),(-,s(s(s(s(0)))),s(s(s(0)))))).
4//2+4-3
?- translate((*,(+,1,2),3)).
(1+2)*3
?- translate((^,(-,(+,s(0),s(s(s(0)))),0),(/,s(s(s(s(0)))),(+,s(0),s(0))))).
(1+3-0)^(4/(1+1))
```

GRUPO II – Programação em Lógica com Restrições (7 valores)

Na resposta às seguintes questões, utilize a biblioteca *CLPFD* do SICStus Prolog.

4. [4 valores]

a) Pretende-se gerar sequências de 5 números que podem tomar valores entre 1 e 9. Implemente um predicado que gere tais sequências tendo em conta as seguintes restrições:

- Não pode haver números pares consecutivos, nem números ímpares consecutivos;
- O número no centro da sequência tem de ser 1 ou 2.
- Não pode haver números repetidos.

3	2	1	4	5

Exemplo:

```
?- seq(Vars).
Vars = [3,2,1,4,5] ?
```

b) Implemente um predicado que resolva uma variante do problema anterior, em que a sequência pode ter dimensão N, tendo em conta as seguintes restrições:

- N tem de ser um número ímpar múltiplo de 3;
- Cada número pode tomar valores entre 1 e 9;
- Um número pode aparecer repetido até no máximo três vezes (o que significa que para determinados valores de N pode não haver solução);
- Não pode haver números pares consecutivos, nem números ímpares consecutivos;
- O número do centro da sequência tem de ser maior que o primeiro elemento e maior do que o último elemento da sequência.

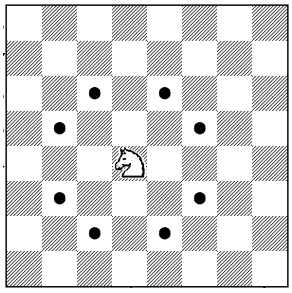
1	2	9	4	9	4	9	6	3

Exemplo para N=9:

```
?- seqn(Vars, 9).
Vars = [1,2,9,4,9,4,9,6,3] ?
```

c) Indique o conjunto de *flags* a passar como argumento ao *labeling* tendo em vista maximizar a eficiência de execução para o problema da alínea b). Justifique, e caso passe mais do que uma *flag* indique qual é a mais crítica.

5. [3 valores]



No xadrez, um cavalo movimenta-se 2 casas num dos eixos (X ou Y) e uma casa no outro eixo (a figura ilustra as casas válidas que o cavalo pode visitar a partir da casa em que está). Se cada casa do tabuleiro tiver um número inteiro indicando o seu valor, pretende-se que seja calculado o caminho que o cavalo pode percorrer (executando jogadas válidas) que obtenha o maior valor possível. O valor do caminho equivale à soma de todas as casas onde o cavalo pousa, sendo que o cavalo não pode pousar duas vezes na mesma casa. Assuma que o tabuleiro tem sempre dimensão 8x8 e que o tamanho do caminho a percorrer pelo cavalo é dado. A casa de partida pode ser qualquer uma. Assuma também que existe um facto ***board_values(V)***, em que ***V*** é uma lista com 64 elementos contendo os valores das casas do tabuleiro (da esquerda para a direita e de cima para baixo, sendo que os primeiros 8 valores correspondem aos valores da primeira linha, os segundos 8 valores correspondem aos valores da segunda linha, e assim sucessivamente). A casa do canto superior esquerdo tem coordenadas (1,1), e a título de exemplo, o cavalo da figura está nas coordenadas (4,5).

Implemente um predicado que permita obter o melhor trajeto possível, dado o número de jogadas a efetuar.

Exemplo de uma possível execução para um caminho de tamanho 3:

```
?- board_values(V).
V =
([3,5,7,7,2,6,2,7,2,4,6,3,9,4,8,6,2,1,8,9,5,9,3,4,5,1,4,8,8,4,7,8,4,3,8,9,
7,8,8,4,1,7,9,8,3,3,8,5,3,2,4,5,2,6,1,2,2,7,1,1,8,1,7,2]).
?- horse(X,Y,TotalValue,3).
X=[5,3,4]
Y=[2,3,5]
TotalValue = 26
```

Interpretação: O cavalo parte da casa (5,2), vai para (3,3) e de seguida para (4,5), totalizando 26 pontos.