Prova com consulta. Duração: 2h30m. Exame da Época de Recurso

Notas:
a) Responda a cada questão numa folha de exame separada (5 questões = 5 folhas).
b) No Grupo I use apenas Prolog Standard, e no Grupo II use também a biblioteca *CLPFD* do SICStus Prolog.
c) Predicados solicitados em alíneas anteriores podem ser utilizados em alíneas seguintes, mesmo que não os tenha implementado.

GRUPO I – Programação em Prolog

1. [**5 valores**] Uma base de dados contém factos como os seguintes, relacionados com sequelas cinematográficas:

```
% sequela(Título,Id,Tipo)
sequela('The Lord of the Rings',1,'fantasia').
sequela('Batman',2,'ficção científica').

% filme(IdSequela,Subtítulo,Ano,Personagens_e_Atores).
filme(1,'The Fellowship of the Ring',2001,
      [aragorn-'Viggo Mortensen',frodo-'Elijah Wood']).
filme(1,'The Two Towers',2002,
      [aragorn-'Viggo Mortensen',gandalf-'Ian McKellen']).
filme(1,'The Return of the King',2003,
      [aragorn-'Viggo Mortensen',frodo-'Elijah Wood']).
filme(2,'-',1989,
      [batman-'Michael Keaton',joker-'Jack Nicholson']).
filme(2,'Batman Returns',1992,
      [batman-'Michael Keaton',catwoman-'Michelle Pfeiffer']).
filme(2,'Batman Forever',1995,
      [batman-'Val Kilmer',robin-'Chris ODonnell',riddler-'Jim Carrey']).
```

a) O predicado ***filme_de(+Sequela,-Subtitulo,-Ano)*** obtém, por *backtracking*, filmes de uma sequela dada, identificada pelo seu título. Implemente-o.

b) O predicado ***entrou_na_sequela(+Ator,-Sequela)*** obtém, por *backtracking*, sequelas (identificadas pelo título) em que um ator dado participou (entrou num ou mais dos seus filmes). O predicado sucede apenas uma vez para cada sequela. Implemente-o

c) O predicado ***atores(+Subtítulo,-ListaAtores)*** obtém, para um filme (subtítulo), uma lista com os nomes dos atores (sem personagens) que nele entraram. Implemente-o.

```
| ?- atores('Batman Forever',LA).
LA = ['Val Kilmer','Chris ODonnell','Jim Carrey']
```

d) O predicado ***manteve_ator(+Sequela,+Personagem)*** sucede se a personagem dada foi representada sempre pelo mesmo ator em todos os filmes da sequela, identificada pelo seu título. Implemente-o. Nota: em caso omisso assume-se que foi o mesmo ator (por exemplo, a personagem ***gandalf*** manteve o mesmo ator nos três filmes).

e) O predicado mostra_sequela(+Sequela) imprime no ecrã os dados relacionados com uma sequela, identificada pelo seu título, na forma exemplificada em baixo. Implemente-o.

```
The Lord of the Rings (fantasia)
   The Fellowship of the Ring (2001)
      aragorn: Viggo Mortensen
      frodo: Elijah Wood
   The Two Towers (2002)
      aragorn: Viggo Mortensen
      gandalf: Ian McKellen
   The Return of the King (2003)
      aragorn: Viggo Mortensen
      frodo: Elijah Wood
```

2. [**4 valores**] O governo de Pequena Utopia mantém registos da sua população em factos Prolog com os seguintes esquemas:

```
nascimento(IdPessoa,IdMãe,Data)
óbito(IdPessoa,Data)
casamento(IdConjuge1,IdConjuge2,Data)
divórcio(IdPessoa1,IdPessoa2,Data)
imigrou(IdPessoa,Data)
```

Os IDs de pessoa são números inteiros. O fundador de Pequena Utopia tem *Id* = 1, e foi um imigrante. Os imigrantes não têm registo de nascimento (um imigrante é uma pessoa que entrou no país para nele passar a residir). As datas são representadas por termos *data(Ano,Mês,Dia)*, e.g. *data(1852,11,27)*.

Nas questões que se seguem, deve implementar o predicado indicado.

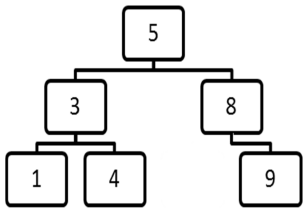
a) Calcule a ***quantidade_de_registos_de_casamento(?N)*** (número de factos).

b) Identifique o ***imigrante_mais_recente(?IdPessoa)*** de Pequena Utopia.

c) Calcule a ***população_ativa(?N)***: quantidade de habitantes (vivos) com idade superior a 14 anos (na data de hoje). Em Pequena Utopia, como os imigrantes não têm registo de nascimento, convencionou-se, para efeitos estatísticos, que 50% são ativos, e 50% não.

d) Determine o ***pai(+IdPessoa,-IdPai)***, assumindo que o pai dum utopiano era o marido da mãe nove meses antes do nascimento.

3. [**4 valores**] Uma árvore binária de pesquisa é uma estrutura de dados em que cada nó tem como descendentes 2 (ou menos) subárvores. Cada nó guarda um valor X: valores inferiores a X estão na sua subárvore esquerda, e valores superiores a X estão na sua subárvore direita. Neste exercício, um nó é sempre representado por uma lista de 3 elementos: [Valor, SubÁrvoreEsquerda, SubÁrvoreDireita]. Caso um nó não tenha algum dos descendentes (e.g. o nó com valor 8 no exemplo seguinte), tal é representado com uma lista vazia. Uma árvore é representada pelo nó da sua raiz, uma vez que a partir dele é possível aceder a todos os nós da árvore. Uma árvore vazia é representada por uma lista vazia. Exemplo:




```
[5,
   [3,
      [1, [], []],
      [4, [], []]
   ],
   [8,
      [],
      [9, [], []]
   ]
]
```

a) Um novo nó é sempre inserido como uma folha da árvore. A inserção deve garantir que a árvore continua a ser uma árvore binária de pesquisa. Implemente o predicado ***adiciona/3*** que insira um valor numa árvore, retornando uma nova árvore contendo esse valor. Utilize e assinale um cut vermelho na sua implementação. Exemplos:

```
| ?- adiciona(6, [5,[3,[1,[],[]],[4,[],[]]],[8,[],[9,[],[]]]], A).
A = [5,[3,[1,[],[]],[4,[],[]]],[8,[6,[],[]],[9,[],[]]]]
| ?- adiciona(5, [], A).
A = [5,[],[]]
```

b) Implemente o predicado ***nega/2*** que dada uma árvore binária de pesquisa A retorna a árvore binária de pesquisa B cujos elementos têm valor simétrico aos elementos de A. A árvore B tem de continuar a ser uma árvore binária de pesquisa. Exemplo:

```
| ?- nega([5,[3,[1,[],[]],[4,[],[]]],[8,[],[9,[],[]]]], B).
B = [-5,[-8,[-9,[],[]],[]],[-3,[-4,[],[]],[-1,[],[]]]]
```

Prova com consulta. Duração: 2h30m. Exame da Época de Recurso

GRUPO II – Programação em Lógica com Restrições

4. [**4 valores**] Pretende-se fazer um gerador de puzzles de Sudoku válidos, com dimensão NxN, isto é, puzzles semelhantes ao que se apresenta na figura (caso de 4x4). As regras para este puzzle são as seguintes:

1	2	3	4
3	4	1	2
2	1	4	3
4	3	2	1

- O número N é um quadrado perfeito.
- Cada linha não pode ter números repetidos.
- Cada coluna não pode ter números repetidos.
- Cada bloco (quadrado de 2x2 no caso do puzzle de 4x4, que tem 4 blocos) não pode ter números repetidos.

a) Considere que já tem os seguintes predicados:

getLine(+LineNumber,+Sudoku,-Line): obtém a linha ***Line*** (uma lista) número ***LineNumber*** do ***Sudoku***.

getColumn(+ColumnNumber,+Sudoku,-Column): obtém a coluna ***Column*** (uma lista) número ***ColumnNumber*** do ***Sudoku***.

getBlock(+BlockNumber,+N,+Sudoku,-Block): obtém o bloco ***Block*** (uma lista de linhas que são listas) de índice ***BlockNumber*** do ***Sudoku*** de dimensão ***NxN***.

Implemente um predicado para gerar puzzles válidos com dimensão NxN. Exemplo:

```
| ?- sudoku(4,S).
S = [[1,2,3,4],[3,4,1,2],[2,1,4,3],[4,3,2,1]] ? ;
S = [[1,2,3,4],[3,4,1,2],[4,1,2,3],[2,3,4,1]] ?
```

b) Acrescente ao problema anterior a restrição adicional de existirem exatamente k 1s numa (e só numa) das diagonais principais do puzzle, onde $k = \sqrt{N}$ (ver figura). Considere que já tem os predicados para obter cada uma das diagonais: ***getDiagonal1(+N,+Sudoku,-Diagonal)*** e ***getDiagonal2(+N,+Sudoku,-Diagonal)***, onde ***Diagonal*** é uma lista. Exemplo:

```
| ?- sudoku2(4,S).
S = [[1,2,3,4],[3,4,2,1],[4,3,1,2],[2,1,4,3]]
```

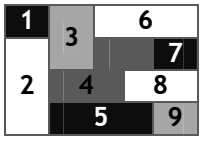
1	2	3	4
3	4	**2**	1
4	3	**1**	2
2	1	4	3

c) Indique que *flag(s)* deverá(ão) ser passadas como argumento ao *labeling* tendo em vista maximizar a eficiência de execução para o problema da alínea b). Justifique a sua escolha.

5. [**3 valores**] Considere o problema de colorir um conjunto de N regiões. Cada cor é identificada por um inteiro diferente ≤ N. É requerido que:

- Regiões adjacentes tenham cores diferentes;
- Exatamente 3 adjacências tenham cores das regiões adjacentes com valores pares;
- O custo total da atribuição das cores seja o mínimo possível, sendo que o custo de atribuir uma cor a uma região é igual a N - Cor + 1.

Implemente o predicado ***regioes/3*** que dados o número de regiões a colorir e uma lista de adjacências determine a cor de cada região. Exemplo:




```
| ?- regioes(9, [1-2, 1-3, 2-3, 2-4, 2-5, 3-4, 3-6, 4-5, 4-6,
4-7, 4-8, 5-8, 5-9, 6-7, 7-8, 8-9], Cores).
Cores = [9,8,7,6,9,8,9,8,7]
```