Prova com consulta. Duração: 2h30m. Exame da Época Normal

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**] Considere a seguinte base de factos:

```
% prof_risco(Pessoa,Profissão)
prof_risco(rui,professor).
prof_risco(to,bombeiro).
prof_risco(marta,bombeiro).
prof_risco(hugo,fuzileiro).
```

```
% fuma_muito(Pessoa)
fuma_muito(sofia).
fuma_muito(rui).
fuma_muito(marta).
fuma_muito(pedro).
```

```
% dorme(Pessoa,Horas)
dorme(pedro,5).
dorme(joao,4).
dorme(marta,5).
dorme(sofia,8).
dorme(hugo,4).
```

a) Implemente o predicado ***dorme_pouco(?P)*** que sucede se a pessoa ***P*** dorme menos de 6 horas por noite.

b) Implemente o predicado ***morre_cedo(?P)*** que sucede se a pessoa ***P*** exerce uma profissão de risco, ou se ***P*** fuma muito e dorme pouco.

c) Apresente a árvore de pesquisa para a pergunta ***?- morre_cedo(P)***. Identifique todas as soluções encontradas.

d) Implemente o predicado ***desgraçado(?P)*** que sucede se ***P*** é a única pessoa com a sua profissão de risco.

e) Implemente o predicado ***listagem/0*** que imprime no ecrã uma listagem de pessoas, sem repetições, da seguinte forma (para a base de dados fornecida):

Nome	Profissão	FumaMuito	DormePouco	MorreCedo
rui	professor	x		x
to	bombeiro			x
marta	bombeiro	x	x	x
hugo	fuzileiro		x	x
sofia	-	x		
pedro	-	x	x	x
joao	-		x	

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. 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_nascimento(?N)*** (número de factos).

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

c) Calcule o indicador demográfico ***população_total(+Data,?QtPessoas)***, i.e. o número de pessoas vivas (nativas ou imigrantes) numa determinada data.

d) O bondoso governo de Pequena Utopia pretende ajudar as ***mães_solteiras(?ListaIds)***, i.e. mães de menores de 18 anos que não estão casadas. Obtenha as suas identificações.

3. [**4 valores**] Considere uma cadeia arbitrária de algarismos, por exemplo 118. Agrupando os algarismos iguais, leríamos "dois uns e um oito", o que pode por sua vez ser representado pela cadeia 2118. Esta depois de lida será convertida em 122118, e assim sucessivamente.

a) Explique o que faz o seguinte predicado, e diga se o *cut* utilizado é verde ou vermelho, justificando (com exemplos):

```
a([X],Xs,[[X|Xs]]).
a([X,X|R],Xs,L2) :- !, a([X|R],[X|Xs],L2).
a([X,Y|R],Xs,[[X|Xs]|RL]) :- a([Y|R],[],RL).
```

b) Tirando partido do predicado anterior, implemente o predicado ***leitura_de(?L,+N)***, em que ***N*** é uma lista de algarismos e ***L*** será uma "leitura" possível dessa lista, ou uma leitura de uma leitura, e assim sucessivamente. O argumento ***L*** pode ou não ser fornecido. Assuma que uma leitura será sempre uma lista de tamanho não inferior à lista que representa a leitura anterior (embora possa ter tamanho inferior ao de um número dado, ver exemplo). Exemplos:

```
| ?- leitura_de([2,1,1,8],[1,1,8]).
yes
| ?- leitura_de(L,[1,1,8]).
L = [2,1,1,8] ? ;
L = [1,2,2,1,1,8] ? ;
L = [1,1,2,2,2,1,1,8] ?
yes
```

```
| ?- leitura_de(L,[4,4,4,4,1,1,1,1]).
L = [4,4,4,1] ? ;
L = [3,4,1,1] ? ;
L = [1,3,1,4,2,1] ?
yes
```

GRUPO II – Programação em Lógica com Restrições

4. [**4 valores**] O Problema de Langford

a) Pretende-se obter sequências de comprimento 6 dos algarismos 1, 2 e 3, que cumpram as seguintes restrições:

- Cada algarismo ocorre duas vezes na sequência.
- Há um algarismo entre os 1's, dois algarismos entre os 2's e três algarismos entre os 3's.

Construa um predicado que permita obter estas sequências, utilizando programação em lógica com restrições. Exemplo:

```
| ?- lf(L).
L = [2,3,1,2,1,3] ? ;
L = [3,1,2,1,3,2] ? ;
no
```

b) Generalizando, dado um número *N*, pretende-se determinar uma sequência de *2*N* números que contenha, para todo o $k \in [1,N]$, uma sequência $S_k = k,\ldots,k$ começada e terminada com o número *k* e com *k* outros números de permeio. Construa o predicado ***langford/2***, de acordo com o seguinte exemplo:

```
| ?- langford(4,L).
L = [2,3,4,2,1,3,1,4] ?
yes
| ?- langford(7,L).
L = [1,4,1,5,6,7,4,2,3,5,2,6,3,7] ?
yes
```

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.

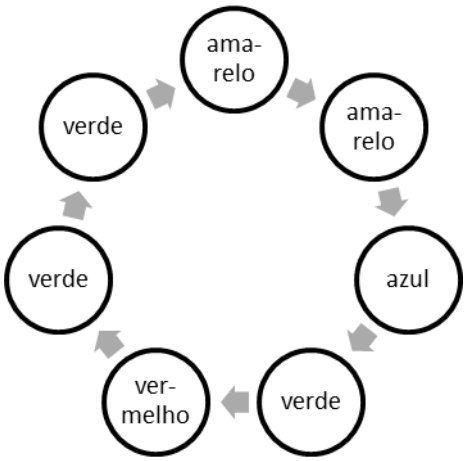
5. [**3 valores**] Considere o problema de gerar uma sequência cíclica de cores. Cada elemento tem uma cor, podendo ser amarelo, azul, verde ou vermelho. As sequências têm sempre 7 ou mais elementos sendo o número de elementos um argumento de entrada. As seguintes restrições devem ser respeitadas:

a) O número de elementos amarelos deve ser superior ao número de elementos azuis;

b) O número de elementos verdes na sequência tem que ser exatamente três;

c) Todos os elementos amarelos presentes na sequência têm que estar juntos;

d) A subsequência azul-verde-vermelho tem que aparecer pelo menos uma vez na sequência.



O custo de uma sequência depende das cores dos seus elementos. A tabela que se segue define o custo por elemento em função da sua cor.

Cor	amarelo	azul	verde	vermelho
Custo	**3€**	**1€**	**5€**	**2€**

Implemente um predicado que, dado o número de elementos, gere uma sequência cíclica de cores que minimiza o custo total. Tendo em vista simplificar o problema e melhorar a eficiência do processo de minimização, implemente também uma ou mais restrições que permitam eliminar soluções simétricas (i.e. soluções equivalentes que podem ser obtidas através da rotação de uma sequência válida).

Exemplo de uma possível execução em que deve ser gerada uma sequência com 7 elementos:

```
| ?- seq(7, Cores, Custo).
Cores = [amarelo,amarelo,azul,verde,vermelho,verde,verde],
Custo = 24 ?
```