Prova 1 - IA (20/10/2015)

Questão 1 (20 pontos)

Considere o programa em Prolog abaixo:

```
f(1, um).
f(s(1), dois).
f(s(s(1)), tres).
f(s(s(s(X))), N) :- f(X, N).
```

Determine se o Prolog vai responder **yes** ou **no** para os cálculos abaixo. Caso ele responda **yes**, dê também o resultado da instanciação para cada variável:

```
?- f(s(1), A).
?- f(s(s(1)), dois).
?- f(s(s(s(s(s(s(1)))))), C).
?- f(D, tres).
```

Questão 2 (25 pontos)

Construa um predicado recursivo em Prolog, denominado **tem_intersecao**, que representa uma relação binária capaz de determinar se dois conjuntos possuem ao menos um elemento comum. O comportamento do predicado é expresso abaixo.

```
?- tem_intersecao([1,2,3], [7,2]).
yes

?- tem_intersecao([1,2,3], [4,5]).
no
```

No caso de uso de outro predicado auxiliar, o mesmo deve ser difinido junto com a resposta desta questão.

Questão 3 (25 pontos)

Construir um predicado em Prolog, denominado **tem_ordem_trocada**, o qual expressa uma relação unária sobre uma lista que sempre possui itens numéricos e alfabéticos. Na referida lista, os itens alfabéticos podem ocorrer intercalados com os itens numéricos ou não. Adicionalmente, a ordem dos itens alfabéticos não afeta o valor-verdade do predicado **tem_ordem_trocada** mas a ordem dos numéricos, sim, nas seguintes situações: (*a*) quando algum item numérico ocorrer fora da ordem crescente, o predicado **tem_ordem_trocada** é verdadeiro; (*b*) quando itens numéricos ocorrerem rigorosamente na ordem crescente, o predicado **tem_ordem_trocada** é falso. Seu comportamento é o expresso abaixo.

```
?- tem_ordem_trocada([a,b,3,c,d,e,4,f,5,g,h,i,8,p]).
no

?- tem_ordem_trocada([a,b,3,c,d,e,4,f,2,g,h,i,5,p]).
yes
```

Para facilitar a solução, assuma a existência de dois predicados, **e_um_numero** e **pertence_a**, e use-os na definição do predicado **tem_ordem_trocada**.

O comportamento do predicado **e_um_numero** está expresso abaixo.

```
?- e_um_numero(53).
yes

?- e_um_numero(alexandre).
no
```

O comportamento do predicado **pertence_a** está expresso abaixo.

```
?- pertence_a(X, [a,1,b,2]).
X = a ? ;
X = 1 ? ;
X = b ? ;
X = 2 ? ;
no
```

Dica: não esqueça que o predicado **pertence_a** é idêntico ao que vimos nos slides, por isso, permite retroação (*backtracking*) inter-cláusulas tal qual foi mostrado no exemplo acima.

Questão 4 (30 pontos)

Considere o domínio de problemas de transformações de estados constituído de uma base com 5 (cinco) lacunas, duas peças pretas e duas peças brancas (uma lacuna fica sempre vazia). Os estados abaixo são exemplos de configurações antes e depois de um movimento singular:

Estado antes

	p	b	p	b

Estado depois

b	p		p	b

Os únicos movimentos singulares possíveis são os seguintes:

- Deslizar uma peça para a lacuna vazia, se ela for adjacente;
- Saltar com uma peça por cima de apenas uma outra peça para ocupar a lacuna vazia.

Construir um predicado denominado **proximo** o qual representa uma relação binária entre um estado antes de uma alteração singular (primeiro termo) e um estado depois dela (segundo termo). O comportamento dele é o expresso abaixo:

```
?- proximo([b, b, v, p, p], Novo).
Novo = [b, b, p, v, p] ?;
Novo = [b, v, b, p, p] ?;
Novo = [b, b, p, p, v] ?;
Novo = [v, b, b, p, p] ?;
no
```

Boa sorte!