Folha de exercicios 3

1. Definir o predicado max, min entre dois números naturais.
2. Escreve o predicado mod (resto da divisão inteira) entre dois números naturais.
3. Defina máximo divisor comum entre dois números naturais a custa de mod.
4. Defina mínimo multiplo comum entre dois números naturais.
5. Escreva o predicado fib e factorial.
6. Escrever um predicado que defina validade de uma frase em lógica proposicional. Aos elementos atómicos da frase são atribuidos os valores lógicos através do predicado verdade. Por exemplos para dizer que a formula A tem valor verdade escrevemos verdade(A). Exemplo do uso
   ?valida((A $\leftarrow$ B) $\rightarrow$ (A $\vee$ B)).

Listas

7. Escrever um predicado que determine se uma dada sublista é um prefixo de outra lista. Exemplo $[1, 2]$ é um prefixo de [1, 2, 3, 4, 5, 6].
8. Definir o mesmo predicado anterior mas usando o predicado append.
9. Definir um predicado consecutivo(X, Y, Ls) que verifica se os elementos X e Y são consecutivos na lista Ls.
10. Escrever um predicado que divida uma dada lista em duas sublistas que contenham, respectivamente, os elementos maiores e menores que um dado numero.

1. Escrever um programa que dada um lista e um elemento retorna uma lista que seja a lista dada com todas as ocorrências do elemento dado removidas.

2. Escrever um predicado que inverta uma lista.

3. Definir um predicado que dado uma palavra na forma de uma lista verifique se esta é um *palindroma* e.g. MADAM é um palindroma.

4. Escrever um programa que determine caminhos num grafo mas que, para além de responder com a ligação possível, responda com o respectivo caminho. Exemplo, para a interrogação ?path(a, X, P) obtinha a resposta X = d, P = [a, b, c, d].

5. Escrever um predicado em Prolog que implemente o algoritmo de ordenação *insertion sort*.

6. Escrever outro predicado que implemente o algoritmo de ordenação QuickSort.

7. Pretende-se implementar o famoso jogo da vida (versão unidimensional). A ideia é implementar o predicado proxima_geracao/2 que dado uma lista representativa da geração actual retorne a lista da geração seguinte. '0' denomina célula morta e '1' viva. As regras de evolução são as seguintes:
   - se houver um '0' com outro '0' a sua direita não se altera essa célula.
   - se houver um '0' com um '1' à sua direita, então o '0' passa a '1'.
   - se houver um '1' com um '0' à sua direita, não se altera essa célula.
   - se houver um '1' com um '1' à sua direita, então o '1' passa a '0'.

Árvores Binárias

1. Definir vários predicados para cada uma das travessias de árvores binárias (pré, in e pos-ordem). A travessia deve estar representada numa lista.

2. Escreva um predicado que verifique se um dado elemento é membro de uma dada árvore binaria.

3. Definir um predicado para inserção ordenada numa árvore e.g. insere(X, T, TA) inser o elemento X na árvore T devolvendo a nova árvore TA.

4. Definir um predicado que coleccione numa lista o conjunto dos elementos folha (terminais) de uma dada árvore binária.

5. Definir um predicado para subarvore(SA, A) em que SA é uma sub-árvore de A.

6. Escreva um predicado nivel(A, N, L) em que dado uma árvore binária A produza uma lista L com os elementos de nivel N da árvore.

7. Em árvores binárias cujos nodos sejam números podemos ter a definição de árvore *equilibrada*:

   `Uma árvore binária diz-se` ***equilibrada*** `se para cada nodo, exceptuando os nodos folha, o número do nodo é igual à soma dos números dos dois nodos filhos.`

   Escreva um predicado equi que dado uma árvore binária verifica se esta é equilibrada ou não. Considere as seguintes queries como exemplos:

   ```
   ?equi(t(7, nil, nil))
     yes.

   ?equi(t(5, t(2, nil, nil), t(2, nil, nil)))
     no.
   ```