Exercícios com aritmética

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 múltiplo 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 atômicos da frase são atribuídos os valores lógicos através do predicado verdade . Por exemplos para dizer que a formula A tem valor verdade escrevemos $\text{verdade}(A)$. Exemplo do uso $?\text{valida}((A \leftarrow B) \rightarrow (A \vee B))$.

Exercícios com 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 $\text{consecutivo}(X, Y, Ls)$ 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 *palíndroma* e.g. MADAM é um palíndroma.
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'.
8. Um homem chega a uma paragem de autocarro às 9H00 de um dia e ai permanece 60 minutos (até às 9H59). Durante este período ele anotou o momento de chegada de cada autocarro que ai pára (registra o minuto). Os autocarros podem pertencer a carreiras diferentes e podem chegar no mesmo momento. No entanto autocarros da mesma carreira chegam sempre a intervalos regulares. Sabe-se ainda que todas as carreiras pararam nessa paragem pelo menos duas vezes.

Escreva um programa em Prolog que dada a lista ordenada com os minutos de paragem de cada autocarro observado, indica um conjunto de carreiras em que eles podem ser agrupados, identificando a respectiva frequência (i.e. o intervalo de tempo entre cada passagem). O resultado deve ser apresentado na forma de uma lista de pares em que o primeiro elemento representa o minuto da primeira paragem e o segundo a frequência de passagem.

Como exemplo a query

```
?agrupa([0, 4, 5, 7, 10, 20, 20, 24, 30, 35, 37, 40, 44, 50, 50], Car)
```

```
Car = [(0, 10), (4, 20), (5, 15), (7, 30)]
```

o que significa que o primeiro autocarro parou ao minuto 0 e pára todos os 10 minutos, o segundo parou ao minuto 4 e pára todos os 20 minutos (parou aos 4, 24 e 44) etc etc.

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 binária.
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 nível 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.
```