Departamento de Matemática da Universidade de Coimbra		
2002/2003	**Algoritmos e Estruturas de Dados II**	**Folha 1** (v1.3)

Estruturas Dinâmicas de Dados

1. Escreva uma função que:
   (a) procure um dado elemento numa lista e devolva `NULL` caso não encontre esse elemento;
   (b) dada uma lista, devolva um ponteiro para o nó com o «*maior*» elemento e o retire da lista.

2. Escreva um sub-programa para:
   (a) ordenar por ordem ascendente uma dada lista;
   (b) ordenar por ordem descendente uma dada lista.

3. Elabore um sub-programa que faça a fusão ordenada de duas listas previamente ordenadas:
   (a) devolvendo uma nova lista;
   (b) devolvendo uma das listas dadas actualizada.

4. O ficheiro de texto `casos.dat` contém um conjunto de nomes, sem nenhuma ordem em especial, aos quais está associada informação de um determinado tipo. Esta informação adicional está registada noutro(s) ficheiro(s).

   Pretende-se representar esses nomes, e a informação que lhes está associada, numa lista. Cada nó da lista permite armazenar uma cadeia de caracteres (i.e., um nome), um registo (do tipo `info`) contendo a informação associada e um ponteiro para o nó seguinte na lista.

   (a) Elabore um procedimento para ler o ficheiro `casos.dat` e construir uma lista ligada onde os nomes aparecem pela mesma ordem.
   (b) Com vista à optimização de repetidas consultas a esta lista, e tendo-se verificado que certos nomes requerem consultas muito mais frequentes, foi decidido que estes **deveriam manter-se no início da lista**. Para esse efeito elabore o sub-programa:

   ```
   int consultaAltera( lista *pInicio, char *esteNome, info *reg );
   ```

   que, para além de consultar `esteNome`, deverá alterar a lista de modo a obter-se o resultado pretendido.
   (c) Como terá certamente reparado, uma versão mais eficiente do procedimento anterior consiste em, após cada consulta, recuar o respectivo nó **uma** posição na lista. Implemente esta versão.

5. Considere a estrutura de dados **Lista de Inteiros**, representada usando uma estrutura **duplamente ligada** que permite o acesso directo às extremidades da lista, e onde os elementos da lista são armazenados por **ordem não-decrescente**.

   (a) Proponha uma representação para essa estrutura de dados.

   (b) Escreva um sub-programa que insira ordenadamente um novo elemento na lista.

   (c) Escreva um sub-programa que remova da lista **todas** as ocorrências de um dado número inteiro, caso lhe pertença, e assinale se essa remoção foi ou não efectuada.

6. Uma **Lista de Cartas de Jogar** pode ser representada de acordo com a seguinte declaração:

```
enum naipes { paus, ouros, copas, espadas };

enum valores { dois, tres, quatro, cinco, seis, sete,
               oito, nove, dez, valete, dama, rei, as };

typedef struct carta *lista;

struct carta {
  enum naipes naipe;
  enum valores valor;
  lista prox;
};
```

   (a) Sendo dada uma Lista de Cartas de Jogar, `cartas`, e a indicação de um naipe, `esteNaipe`, elabore um sub-programa para construir a lista ordenada das cartas de `esteNaipe` que nela ocorrem.

   (b) Proponha uma estrutura de dados adequada à representação de uma «Mão» (conjunto de cartas que um jogador possui). Indique as vantagens da estrutura que propôs.

   (c) Elabore um procedimento para converter uma dada Lista de Cartas de Jogar à representação definida na alínea anterior.