Departamento de Matemática da Universidade de Coimbra		
2002/2003	**Algoritmos e Estruturas de Dados II**	**Folha 2** (v1.2)

Tipos Abstractos de Dados

1. (a) Defina o Tipo Abstracto de Dados `POLINOMIO` de uma variável e coeficientes reais, por intermédio de uma escolha adequada de operações.

   (b) Proponha uma estrutura de dados apropriada para a representação desse tipo.

   (c) Implemente a operação:

   `DIVIDIR(pol1, pol2)` ——> quociente, resto.

   (d) Elabore um sub-programa para calcular o máximo divisor comum de dois polinómios.

2. Sabendo que, para $n \geq k \geq 0$,

   $$C_k^n = C_k^{n-1} + C_{k-1}^{n-1}$$

   com $C_n^n = C_0^n = 1$, elabore um procedimento iterativo para calcular C_k^n simulando uma estratégia recorrente.

   Este procedimento, para além do valor C_k^n, deve ainda devolver o número de simulações de chamadas recorrentes efectuadas.

   Sugestão: Utilize o Tipo Abstracto de Dados `PILHA` para registar os sucessivos parâmetros das simulações das chamadas recorrentes.

3. Considere o Tipo Abstracto de Dados `FILA DE CARACTERES`, representado usando uma **lista ligada**.

   (a) Defina o Tipo Abstracto de Dados `FILA DE CARACTERES`. Descreva sucintamente as suas operações.

   (b) Proponha uma representação para esse Tipo Abstracto de Dados.

   (c) Escreva um sub-programa que junte um elemento a uma fila.

   (d) Escreva um sub-programa que retire um elemento de uma fila.

   (e) Escreva um sub-programa **iterativo** que liste os elementos de uma fila pela ordem em que se encontram na fila.

   (f) Escreva, agora, um sub-programa **recorrente** que liste pela **ordem inversa** os elementos de uma fila.

4. Existe uma variante particular do Tipo Abstracto de Dados `FILA` que permite inserir ou remover elementos em qualquer um dos seus extremos e que vamos denominar `FILA DUPLA` (*DEQUE* em inglês).

   Proponha um conjunto mínimo de operações e uma representação que permita implementar o Tipo Abstracto de Dados `FILA DUPLA`.

5. Considere o Tipo Abstracto de Dados *Árvore Binária de Inteiros*, em cujos nós é possível armazenar um número inteiro.

   Considere também que os números inteiros se encontram registados ≪**em-ordem**≫ **crescente**.

   Elabore sub-programas que permitam:

   (a) Indicar o número de folhas de uma árvore dada.

(b) Listar os n primeiros elementos armazenados numa dada árvore.

(c) Inserir, **ordenadamente** e de modo **iterativo**, um dado número inteiro numa árvore, caso não esteja já registado na árvore.

(d) Inserir, **ordenadamente** e, agora, de modo **recursivo**, um dado número inteiro numa árvore, caso não esteja já registado na árvore.

6. Considere o Tipo Abstracto de Dados *Árvore Binária de Inteiros*, em cujos nós é possível armazenar um número inteiro.

   Considere também que os números inteiros **não** se encontram registados em nenhuma **ordem particular**.

   Elabore sub-programas que permitam:

   (a) Listar os elementos registados nas **folhas** de uma árvore.

   (b) Obter um **ponteiro** para o nó que contém o **maior elemento** registado numa árvore.

   (c) Dados uma árvore e dois números inteiros a e b, verificar se b é **descendente** de a — i.e., se b pertence a uma das sub-árvores do nó que contém a.
   Atenção: deverá ser contemplada a possibilidade de a e/ou b não pertencerem à árvore dada.

7. Seja $\mathcal{ABP}$ o tipo abstracto das árvores Binárias de Pesquisa, onde a informação associada a cada nó é identificada por um campo chamado `chave`. Existe, naturalmente, uma relação de ordem total definida no conjunto das *chaves*.

   (a) Escreva uma definição formal do tipo $\mathcal{ABP}$.

   (b) Directamente a partir da definição anterior, demonstre (por indução) que a pesquisa de um elemento numa árvore **total** do tipo $\mathcal{ABP}$ é de ordem de complexidade logarítmica.

   (c) Elabore um procedimento para a **remoção** de eventuais repetições numa árvore do tipo $\mathcal{ABP}$.

8. A informação referente ao aproveitamento dos 100 alunos de um dado curso encontra-se armazenada de acordo com as seguintes declarações:

```
typedef char *nome;

typedef struct cadeira *ponteiro;

struct cadeira {
  nome nomeDeCadeira;
  unsigned short nota;
  ponteiro prox;
};

typedef struct aluno *arvBin;

struct aluno {
  nome nomeDeAluno;
  nome nomeDeCurso;
  ponteiro listaDeCadeiras;
  arvBin esq;
  arvBin dir;
};
```

Assumindo que:

- a `arvBin` definida é uma *árvore binária de pesquisa* onde os vértices estão ordenados alfabeticamente **em-ordem** por `nomeDeAluno`;
- cada `listaDeCadeiras` está ordenada por ordem alfabética de `nomeDeCadeira`;

elabore sub-programas para:

(a) **Listar por ordem alfabética** os nomes dos alunos de `esteCurso`.

(b) **Actualizar** a estrutura de dados de acordo com os resultados obtidos em `estaCadeira`.
A pauta respectiva está registada num ficheiro onde, em cada linha, se encontra o nome de um aluno (50 caracteres), seguido de `':'` e de uma letra (D ou R) ou de um valor inteiro entre 10 e 20.
Se um determinado aluno já tiver aprovação em `estaCadeira`, deverá ser-lhe atribuída a melhor das duas notas.
Note que: cada vértice da árvore só deve ser visitado **uma única vez**.