Departamento de Matemática da Universidade de Coimbra

Métodos de Programação II	Teste Teórico-Prático e Prático Final	18/12/2007

Exercício 1

1.a) Explique o que entende por estratégia de resolução algorítmica da classe:

(i) *dividir para conquistar* (do ingês, *divide–and–conquer*).

(ii) *bactraking*.

1.b) Apresente um **algoritmo genérico de backtracking** para, dado um mapa plano, encontrar todas as possíveis colorações desse mapa com 4 cores.

Nota: apresente exclusivamente o algoritmo principal (que vai pintar um país e decidir quando termina), dedicando apenas mais uma ou duas linhas a outro tipo de instruções que ache estritamente necessário descrever.

1.c) Suponha que tem um digrafo pesado com n nós e m arcos dado por uma lista de adjacências: os nós são representados por uma lista ligada, onde cada elemento tem um apontador para a sua lista ligada de vizinhos; cada vizinho é definido pelo número do nó sucessor a este e o peso do arco que liga os dois nós.

Proponha uma estrutura de dados adequada à representação descrita.

Exercício 2

Suponha que temos lista ligada simples de números inteiros com elementos do tipo *NoLista* e apontadores do tipo *Seta* e que é apontada por uma base do tipo *BASE*, de acordo com as declarações seguintes:

```
typedef struct noLst{
           int valor;
           struct noLst *prox;
        }NoLista, *Seta;
```

```
typedef struct noBase{
            struct noLst *ponta;
            struct noLst *ultimo;
          }BASE;
```

Note que, na *BASE*, o campo *ponta* é um ponteiro para o ínicio da lista e o ponteiro *ultimo* aponta o último elemento da lista.

Suponha ainda que estão definidos os seguintes módulos:

```
BASE Cria_Base() //do nada, cria uma base vazia
BOOL Vazia_B(BASE B) // indica se uma base aponta a lista vazia
Seta Novo_noL(int valor) // devolve uma caixa da lista
BASE InsereFim_L(BASE B, int valor) // insere uma nova caixa no fim da lista;
            // devolvendo a base actualizada
BASE Remove_L(BASE B, Seta este) // retira esta caixa da lista;
            // devolve a base actualizada
BOOL Menor(Seta este, int num) //devolve TRUE sse este->valor <= num
```

2.a) Implemente a sua versão do módulo `BASE InsereFim_L(BASE B, int valor)`.

2.b) Implemente o algoritmo de ordenamento *Quicksort* para uma dada lista L, mas com a seguinte **variante** no algoritmo de separação:

- Caso a lista L tenha 2 ou mais elementos:
  - escolher 2 quaisquer elementos (números) da lista dada, n_1 e n_2;
  - verificar qual é o menor desses elementos (suponha-se que é n_1);
  - separar os elementos de L em 3 listas: L_1 deverá conter todos os elementos com valor menor ou igual a n_1; L_2 deverá conter todos os elementos com valor menor ou igual a n_2 mas superiores a n_1; L_3 deverá conter todos os elementos com valor superior a n_2;

IDENTIFICAÇÃO

Nome Completo:

Número de Aluno:
