	Departamento de Matemática da Universidade de Coimbra		
	Métodos de Programação II	**Exame**	**22/01/2009**

Nota: A Linguagem de Programação utilizada deve ser C e a resposta completa a um exercício de programação deve incluir a escrita do algoritmo a usar em pseudo-código.

1. Considere uma tabela de, no máximo, 100 elementos **inteiros e não–negativos** que deverá ser identificada pelo nome `NUM`.

   (a) Implemente um algoritmo que devolva a soma de todos os elementos de um dado vector `d` do tipo `NUM`.

   (b) Elabore um procedimento para, a partir da leitura do *standard input*, estabelecer a representação decimal de um número $k \in \mathbb{N}_0$ num vector do tipo `NUM`, onde, obviamente, todos os elementos de `d` serão um dos possíveis dez algarismos: $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

   (c) É sabido que qualquer $n \in \mathbb{N}_0$ goza de uma conhecida propriedade que pode ser expressa como: *P(n)* $\equiv$ { *O resto da divisão de n por nove é igual ao resto da divisão por nove da soma de todos os seus dígitos* }.

   Usando a alínea (1a) construa um procedimento para, dado um número natural não–negativo através da sua representação no tipo `NUM`, verificar esta afirmação.

2. Considere as seguintes declarações de tipos:

```
typedef struct student{
          char *nome;
          struct student *prox;
        } Aluno, *PTAluno;
```

   (a) Leia os três sub–programas seguintes e explique qual o efeito de cada um deles, supondo que `ponta` é uma estrutura, já construída, do tipo `PTAluno` (ou seja, quais os resultados em função dos dados).

```
void XX(PTAluno ponta, char *ident)
{   ponta = NULL; ponta->nome = ident; ponta->prox NULL; }

int XY(PTAluno ponta, char *ident)
{ PTAluno p;
  int c, continua;
    p = ponta;  c = 0; continua = 1;
    while( (p != NULL) && continua )
    {  c = c + 1;
       if( strcmp(p->nome,ident) == 0 ) continua = 0;
       p = p->prox;
    }
    return c;
}

void YY(PTAluno *ponta)
```

```
{ PTAluno este;
    while( *ponta )
    {
      este = *ponta; *ponta = (*ponta)->prox;
      free(este);
    }
}
```

(b) Escreva um sub–programa para, dados *k*, *estenome* e uma lista de nomes, *ponta*, do tipo `PTAluno`, procurar e remover o elemento que está na *k*-ésima posição dessa lista *desde que esse elemento contenha estenome*.

Note que pode usar ou modificar qualquer um dos módulos da alínea anterior se achar necessário.

3. Considere as seguintes declarações de tipos:

```
#define Limite  = 100;
#define NumReal = 5;
#define NumIm   = 6;
typedef struc cplx{ double re, im; }Complexo;
typedef struc nm{
          int tiponum;
          union{ double real; Complexo imaginario;
          } num;
        }Numeros;
typedef Numeros Lista[Limite];
```

(a) Que tipo de informação representam as declarações acima?

(b) Escreva um procedimento para, dados dois "números" do tipo `Numeros`, efectuar a sua **soma**, devolvendo o resultado num registo do mesmo tipo.

(c) **Modifique**, no que for necessário, as declarações apresentadas, de modo a alterar o tipo `Lista` de tabela de `Numeros` para lista ligada linear de `Numeros`.

(d) Indique quais as vantagens e quais as desvantagens decorrentes da alteração efectuada.

(e) Utilizando o módulo construído na alínea (3b) e a **nova declaração de** `Lista`, elabore um novo módulo para efectuar a **soma de todos os elementos** de uma dada `Lista` de `Numeros`, e devolver o valor obtido.

(f) Relembre o algoritmo do *Ordenamento por Inserção Linear* "a partir do início de uma sequência, ver o próximo elemento: se estiver desordenado, colocar na subsequência anterior já ordenada; senão avançar a subsequência já ordenada para este elemento".

Implemente esta estratégia para **ordenar uma dada** `Lista` de `Numeros` que só contém números reais.