	Departamento de Matemática da Universidade de Coimbra		
	Métodos de Programação II	**Exame de Recurso**	**02/02/2009**

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 **do tipo carácter** que deverá ser identificada pelo nome `Frase`.

   (a) Implemente um algoritmo que remova de todos os elementos do tipo espaço em branco de um dado vector `d` do tipo `Frase`.

   (b) Elabore um módulo para, a partir da leitura do *standard input*, carregar uma linha completa do *standard input*, eliminar todos os espaços em branco e devolver 1 se o resultado final é um número (real ou inteiro), e 0 caso não seja um número.

2. Considere as seguintes declarações de tipos:

```
typedef struct func{
          char *nome;
          struct func *prox;
        } Funcion, *PTFunc;
```

   (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 `PTFunc` (ou seja, quais os resultados em função dos dados).

```
PTFunc XX(PTFunc ponta, char *ident)
{   PTFunc aux;
        aux = (PTFunc) malloc(sizeof(Funcion));
        aux->nome = ident; aux->prox = NULL;
        return aux;
 }

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

void YY(PTFunc *ponta, char *ident)
{ PTFunc este, outro;
      outro = XY(*ponta, ident);
      if( outro != NULL ){
           este = outro->prox; outro->prox = este->prox;  free(este);
```

```
        } else {
              if( strcmp((*ponta)->nome, ident) == 0 ){
                    este = *ponta; *ponta = este->prox;  free(este);
              }
        }
  }
```

(b) Escreva um sub–programa para, dados *k*, *estenome* e uma lista de nomes, *ponta*, do tipo `PTFunc`, procurar e remover todos os elemento com nomes iguais a *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.