Departamento de Matemática da Universidade de Coimbra

Métodos de Programação II	Teste Teórico-Prático 2	9/1/2007

No que se segue, considere sempre listas lineares simples.

Exercício 1

Explique o que está mal no módulo seguinte e proceda às correcções que achar necessárias.

```
/* retira e devolve o ultimo elemento da lista */
    Seta devolvedoFim(Seta ponta)
    { Seta P = NULL; //P: auxiliar para percorrer a lista

      if(ponta){
           P = ponta;
           if(P->prox){ //ponta tem mais do que 1 elemento
               while(P->prox->prox != NULL)
                    P = P->prox;
           }else ponta = NULL; // ponta tem 1 elemento
      }

      return P;
    }/fim devolvedoFim
```

Exercício 2

Recorde o algoritmo dado nas aulas para inserir um novo elemento **em ordem** numa dada lista já ordenada por ordem estritamente crescente:

Dados: uma lista, *ponta*, ordenada por ordem estritamente crescente dos seus elementos e um inteiro, k.

Resultados: A lista com k como novo elemento (caso não existisse já na lista) e ainda ordenada por ordem estritamente crescente.

Algoritmo:

- pesquisar, em ordem, o elemento k na lista, devolvendo um ponteiro, *ant*, para a "caixa" anterior em ordem, caso não exista já na lista;
- se não encontrou, então:
  - pedir nova "caixa" à memória, *novo*, e colocar k como elemento;
  - se *novo* elemento é o primeiro, inserir à cabeça da lista;
  - caso contrário:
    - actualizar campo com o apontador para o elemento seguinte de *novo* com o valor do seguinte a *ant*;
    - actualizar campo com o apontador para o elemento seguinte a *ant* para *novo*;
- devolver a lista (*ponta*).

Descreva um algoritmo para, dada uma lista qualquer, a altere de modo a ordenar os seus elementos por ordem estritamente crescente e devolver essa lista ordenada.

Notas: Se quiser, pode usar um ponteiro do tipo lista auxiliar para construir a lista ordenada e pode supôr que os elementos da lista dada são todos diferentes.

Exercício 3

Considere as seguintes declarações de tipos para implementar uma lista de alunos:

```
#define TRUE 1
#define FALSE 0
Typedef int BOOL;
typedef int Notas;
```

```
typdef struct cx{
          char[60] nome;
          BOOL tipoaval;
          Notas avalia[10];
       } Aluno;
 typedef Aluno[60] Lista;
```

Onde `tipoaval` é TRUE se o aluno for de avaliação contínua e FALSE caso contrário.

3.a) Explique quais as vantagens e as desvantagens entre a implementação de uma lista como uma tabela (estrutura estática) ou como uma lista ligada (estrutura dinâmica).

3.b) Modifique, no que for necessário, as declarações apresentadas, de modo a alterar o tipo `Lista` de tabela para **lista ligada simples** de `Aluno`.