Programação Imperativa
2º Teste

1º Ano – LEI/LCC

20 de Junho de 2013
(Duração: 90 min)

1. Considere o seguinte tipo para representar listas ligadas de inteiros:

```
typedef struct lista {
    int valor;
    struct lista *prox;
} Nodo, *LInt;
```

   Apresente uma definição iterativa da função `int take (int n, LInt *l)` que, dado um inteiro `n` e uma lista ligada de inteiros `l`, apaga de `l` todos os nodos para além do `n`-ésimo (libertando o respectivo espaço). Se a lista tiver `n` ou menos nodos, a função não altera a lista.

   A função deve retornar o comprimento final da lista.

2. Considere o seguinte tipo para representar árvores binárias de inteiros:

```
typedef struct abin {
    int valor;
    struct abin *esq, *dir;
} Nodo, *Abin;
```

   Apresente uma definição da função `int maiores (Abin a, int x)` que, dada uma árvore binária de procura de inteiros e um inteiro, conta quantos elementos da árvore são maiores que o inteiro dado.

   Certifique-se que a sua função toma partido da ordenação da árvore.

3. O comando `wc` (**w**ord **c**ount) do Unix conta os caracteres, palavras e linhas de um ficheiro.

   Apresente uma definição da função `int wc (char *filename)` que dado o nome de um ficheiro, imprime no ecran o número de linhas, palavras e caracteres desse ficheiro.

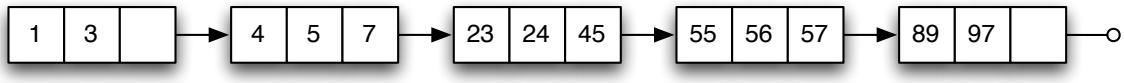
   A função deve retornar o tamanho da maior palavra encontrada no ficheiro.

   Considere que as várias palavras de um texto são separadas por um ou mais caracteres brancos (para os quais a função `isspace` retorna verdadeiro).

4. Uma forma de representar conjuntos de inteiros consiste em usar uma lista de blocos em que cada bloco consiste num array de inteiros **ordenado e sem repetições**. A lista também se encontra ordenada, no sentido em que o primeiro elemento de cada bloco é maior do que o último do bloco anterior. Por exemplo, se quisermos representar o conjunto

   `1, 3, 4, 5, 7, 23, 45, 55, 56, 57, 89, 97`

   e o tamanho de cada array for 3, podemos ter a seguinte lista:



   Considere então a seguinte defínção:

```
#define N ...
typedef struct bloco {
    int quantos; // elementos ocupados
    int valores[N];
    struct bloco *prox;
} Bloco, *LBoco;
```

   (a) Apresente uma definição da função `int pertence (LBloco l, int x)`, que, dado um conjunto representado desta forma, e um inteiro, testa se esse inteiro pertence ao conjunto. A função deve retornar `1` caso o elemento pertença e `0` no outro caso.

   (b) Apresente uma definição da função `int acrescenta (LBloco *l, int n)` que acrescenta um inteiro a um conjunto. A função deve devolver `0` em caso de sucesso.

   Assegure que, caso o conjunto tenha de ser representado em mais do que um bloco, nenhum bloco está menos do que 50% ocupado. Por exemplo, no exemplo acima, em que o tamanho do bloco é 3, nenhum bloco tem menos do que 2 posições ocupadas.