Programação Imperativa
Exame da Época Especial

1º Ano – LEI/LCC

10 de Setembro de 2013
(Duração: 2h)

1. Defina uma função `int sqrtInt (int n)` que, dado um inteiro positivo `n`, calcula o maior inteiro que é menor do que a raíz quadrada de `n`.

   Por exemplo, `sqrtInt (49)` e `sqrtInt (63)` correspondem ambos a `7` uma vez que $7 \leq \sqrt{49} < 8$ e $7 \leq \sqrt{63} < 8$.

2. Defina uma função `void inverte (int v[], int N)` que inverte a ordem por que aparecem os `N` primeiros elementos no vector `v`.

   Por exemplo, se os primeiros 5 elementos do vector `v` forem `{1,2,3,4,5}`, a invocação `inverte(v,5)` deve colocar nas primeiras posições do vector os valores `{5,4,3,2,1}` e não alterar o resto do vector.

3. Considere a seguinte definição de um tipo para representar listas ligadas de inteiros.

```
typedef struct llint {
    int valor;
    struct llint *prox;
} NodoL, *LLint;
```

   Apresente uma definição não recursiva da função `LLint cloneR (LLint)` que cria uma nova lista ligada com os elementos por ordem inversa.

   Por exemplo, se a lista `l` tiver 5 elementos com os valores `{1,2,3,4,5}` por esta ordem, a invocação `cloneR(l,5)` deve corresponder a uma nova lista com os elementos `{5,4,3,2,1}` por esta ordem.

4. Considere a seguinte definição de um tipo para representar árvores binárias de inteiros.

```
typedef struct abint {
    int valor;
    struct abint *esq, *dir;
} NodoAB, *ABint;
```

   (a) Defina uma função `int freeAB (ABint a)` que liberta o espaço ocupado por uma árvore binária, retornando o número de nodos libertados.

(b) Defina uma função `int pruneAB (ABint *a, int l)` que remove (libertando o espaço respectivo) todos os elementos da árvore `a` que estão a uma profundidade superior a `l`, retornando o número de elementos removidos (use, se precisar a função `freeAB` referida atrás).

Assuma que a profundidade da raíz da árvore é 1, e por isso a invocação `pruneAB(&a,0)` corresponde a remover todos os elementos da árvore `a`.

5. A função `char *fgets (char *s, int n, FILE *f)` lê para `s`, a partir do ficheiro `f` uma string com no máximo `n` caracteres. A função devolve o endereço `s` caso não ocorram erros e seja lido pelo menos um caracter. Caso contrário devolve `NULL`.

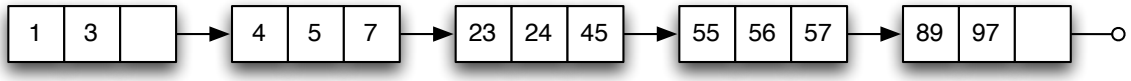
   Essa string pode ter menos do que `n` caracteres caso o ficheiro termine ou apareça o caracter `'\n'`.

   Apresente uma definição desta função.

6. 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, 24, 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 quantos (LBloco l)` que, dado um conjunto representado desta forma, calcula o número de elementos do conjunto.

   No exemplo apresentado a função deveria retornar `13`.

   (b) Apresente uma definição da função `int compacta (LBloco *l)` que reorganiza os números pelos blocos de forma a que todos os blocos (com possível excepção para o último, estão completamente preenchidos.

   A função deve devolver o número final de blocos e libertar todo o espaço não usado.