# Programação Imperativa
# 2º Teste

1º Ano – LEI/LCC

18 de Junho de 2014
(Duração: 90 min)

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

```
typedef struct slist {
    int valor;
    struct slist *prox;
} *LInt;
```

   Defina uma função `LInt fromArray (int v[], int N)` que, dado um array `v` com `N` elementos, ordenado por ordem crescente, constrói uma lista ordenada com os elementos do array, pela mesma ordem.

2. Considere a seguinte alternativa para implementar stacks de números inteiros baseada numa lista ligada de arrays.

```
#define BSize 100
typedef struct larray{
  int valores[BSize];
  struct larray *prox;
} *LArrays;
```

```
typedef struct stack{
  int sp;
  LArrays s;
} Stack;
```

   O campo `sp` armazena o número de elementos do primeiro bloco que estão a ser usados (todos os outros blocos estão completamente ocupados).

   Apresente definições das funções de inserção (`void push (Stack *st, int x)`) e remoção (`int pop (Stack *st, int *t)`) tendo em atenção que

   - Na inserção, caso o primeiro bloco esteja completamente ocupado, é criado um novo que será o primeiro elemento da lista de blocos
   - A remoção retorna `0` em caso de sucesso (stack não vazia) e caso o bloco de onde o elemento é removido fique vazio, remove-o da lista de blocos.

3. Considere os seguintes tipos para representar listas de inteiros e de pares de inteiros:

```
typedef struct spares {
    int x, y;
    struct spares *prox;
} Par, *LPares;
typedef struct slist {
    int valor;
    struct slist *prox;
} Nodo, *LInt;
```

Apresente uma definição não recursiva da função

```
LPares zip (LInt a, LInt b, int *c)
```

que constrói uma lista de pares a partir dos elementos de duas listas de inteiros. A função deverá retornar em `c` o tamanho da lista construída e que corresponde ao comprimento da menor das listas recebidas como parâmetro.

Por exemplo, se `x` for a lista com os elementos `[11,12,13,14,15,16]` e `y` a lista com os elementos `[10,20,30]`, a invocação `zip (x,y,&z)` deve dar como resultado uma lista com `[(11,10),(12,20),(13,30)]` (por esta ordem) e colocar em `z` o valor `3`.

4. Considere o seguinte tipo para árvores, em que os nós também guardam um apontador para o pai:

```
typedef struct no {
    int value;
    struct no *esq,*dir,*pai;
} No, *Tree;
```

   (a) Defina uma função `void calculaPais(Tree t)` que, dada uma árvore (onde os apontadores para os pais ainda não estão calculados) coloca nos campos `pai` os valores correctos nesses apontadores.

   (b) Defina a função `Tree next(Tree t)` que dado um apontador para um nó arbitrário de uma árvore binária de procura devolve o apontador para o nó que o segue numa travessia inorder ou `NULL` se tal não existir.