Programação Imperativa – CC (1º ano)

Teste final

16 de Julho de 2010 (14.30h00)

Dispõe de **2:00 horas** para realizar este teste.

Questão 1 (cálculo numérico) [4v = 2 + 2]

a) Defina uma função `int Zeros (int x)` que, dado um número inteiro calcula quantos dos bits da sua representação binária são 0. Exemplo:

```
...
x = Zeros(6);
printf("%d",x);
...
```

Este excerto de código colocaria o valor 1 no monitor.

b) Defina uma função `int eprimo (int x)` que, dado um número inteiro verifica se este é primo devolvendo 1 em caso afirmativo e 0 no outro caso.

Questão 2 (arrays) [4v = 2 + 2]

a) Defina uma função que recebe como parâmetro uma string e a inverte, devolvendo o apontador para a string invertida;

b) Defina uma função que recebe como parâmetro duas strings, e que dá como resultado um inteiro indicando quantas vezes a segunda string ocorre dentro da primeira.

Questão 3 (pergunta dada) [2v]

Considere uma árvore binária de procura definida pelo seguinte tipo abstracto de dados em C:

```
typedef struct sArvBin
  {
    int valor;
    struct sArvBin *esq, *dir;
  }
    *ArvBin, NodoArvBin;
```

Especifique uma função que faz uma listagem *posorder* dos nodos da árvore (na especificação desta função está proibida a utilização de recursividade quer na função principal quer em qualquer função auxiliar).

Questão 4 (estruturas dinâmicas) [10v = 2 + 1 + 2 + 2 + 1 + 2]

Considere uma árvore genealógica ascendente (que relaciona uma pessoa com os seus pais) e que em termos mais formais pode ser descrita da seguinte maneira:

```
AG = Indivíduo * Pai * Mãe
          | desconhecido

Indivíduo = nome * BI * data-nascimento
Pai = Mãe = AG
```

Com base nesta especificação abstracta desenvolva as seguintes alíneas:

a) Especifique tipos abstractos de dados em C que permitam manipular árvores genealógicas. Sugestão: crie o tipo *Individuo* e o tipo *ArvGenoa*;

b) Especifique uma função de travessia que lista no monitor toda a informação guardada na árvore:

```
void ListArvGenoa( ArvGenoa a);
```

c) Defina uma outra função `IndivList IndivPorNasc(ArvGenoa a);` que, dada uma árvore genealógica, retorna uma lista de indivíduos ordenados por data de nascimento (defina o tipo *IndivList*);

d) Defina uma função `IndivList Avos( ArvGenoa a, BI b);` que, dada uma árvore genealógica o BI de um indivíduo, retorna a lista de indivíduos que são avós do detentor do BI (caso o elemento com esse BI não exista ou não tenha avós deverá retornar uma lista vazia;

e) Defina uma função `int Geracoes( ArvGenoa a);` que calcula o número de gerações representadas na árvore;

f) Defina uma função `IndivList Descendentes( ArvGenoa a, BI b);` que calcula a lista de descendentes do indivíduo cujo BI é passado como argumento;