Programação Imperativa – EI (1º ano)

Exame de Recurso

7 de Julho de 2010 (9h30)

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

Questão 1 (cálculo numérico)

Especifique em C uma função que recebe um argumento do tipo inteiro e positivo, a, e dá como resultado o somatório dos divisores de a.

Apresente definições para duas versões desta função:

a) `somadivit` — que realiza o cálculo de forma iterativa;

b) `somadivrec` — que realiza o cálculo de forma recursiva.

```
int somadivit( int a )
{
    ...
}

int somadivrec( int a )
{
    ...
}
```

Questão 2 (arrays)

a) A função $strcspn(s1, s2)$ calcula o comprimento do segmento inicial da string $s1$ que não contem nenhum dos caracteres presentes em $s2$.

Apresente uma definição em C para esta função:

```
int strcspn( char *s1, char *s2 )
{
    ...
}
```

b) Defina uma função que recebe como parâmetro, dois Polinónios P1 e P2 e retorna 1, se P2 for a derivada de P1, e 0 caso contrário. Considere os seguintes tipos e assinatura da função:

```
#define MAXTERMO 100
typedef struct sTermo
  {
    float coef;
    int exp;
  } Termo;

typedef struct sPolinomio
  {
    Termo pol[MAXTERMO];
    int  ntermos;
  }  Polinomio;

int ederivada( Polinomio p1, Polinomio p2 )
{
    ...
}
```

Questão 3 (pergunta dada)

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 *preorder* 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 (Listas Ligadas)

Considere os seguintes tipos de dados e funções que modelam listas ligadas em C.

```
typedef struct Cel
{ int         elem;
  struct Cel* next;
} Celula , *LISTA;

void showLISTA (LISTA);

typedef struct
{ LISTA a;
  LISTA b;
} DUASLISTAS;

void  showDUASLISTAS (DUASLISTAS);

DUASLISTAS  split (LISTA);
```

Tendo como base esta definição, responda às alíneas seguintes:

a) Escreva em C a função `showDUASLISTAS` que mostra o valor das listas `a` e `b` contidas no tipo `DUASLISTAS` (considere que a função `showLISTA` está já definida).

b) Escreva em C a função `insNodoFim`, que insere um nodo (**não um valor**) no fim de uma lista. O tipo da função é:

```
LISTA   insNodoFim (Celula * , LISTA );
```

c) Escreva a função `split` que recebe como argumento uma lista ligada de inteiros e a divide em duas: uma (a lista `a`) ficará com os número maiores que o valor `10` e a outra (a lista `b`) com os outros elementos. Por exemplo, dada uma lista com o elementos `[2,15,12,7,223]`, a função produzirá duas listas `a = [2,7]` e `b = [15,12,223]`.

Escreva esta função sem criar memória dinâmica adicional e mantendo a ordem relativa dos elementos na lista original. (para resolver este exercício considere a utilização da função anterior).

Questão 5 (estruturas dinâmicas)

No teste final de 15 de Junho último, foi proposto um exercício que consistia na definição de uma árvore genealógica ascendente (que relaciona uma pessoa com os seus pais) e que foi descrita da seguinte forma:

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

Pai = Mãe = AG
```

Em que o tipo `Indivíduo` representava a informação de uma pessoa. Considere agora que se pretende desenvolver uma aplicação para a *Associação Portuguesa do Cão Da Serra da Estrela* (`http://www.apcse.com.pt/`) de modo a permitir criar, validar e manter um registo de *pedigree* dos cães desta raça. O *pedigree* prova a pureza de raça dum cão. Considera-se que um cão tem *pedigree* se:

- Os pães e os avós do cão são conhecidos e *todos* têm *pedigree*;
- Por decisão de um júri num concurso que atesta a pureza do cão. Neste caso os pais e avós podem não ser conhecidos e/ou não ter *pedigree*.

Com base nesta descrição desenvolva as seguintes alíneas:

a) Especifique tipos abstractos de dados em C que permitam manipular árvores genealógicas para os cães Serra da Estrela. Considere que a informação que é registada sobre um cão é o seu nome, número de registo (que funciona como o número de BI para os cães) e o facto de o cão ter *pedigree* ou não. Sugestão: defina os tipos $Individuo$ e AG;

b) Escreva uma função em C, com nome `temPedigree`, que dado o número de registo do cão, indica se o cão tem *pedigree* ou não. (esta função apenas devolve o valor do campo que regista o *pedigree*; não testa a sua validade!);

c) Escreva uma função em C, com nome `validaPedigree`, que verifica se o *pedigree* dos cães na árvore geneológica está de acordo com as regras apresentadas. A função tem tipo:

```
typedef int BOOL;
BOOL validaPedigree(AG);
```

Note que para a árvore estar correcta não pode existir um cão sem *pedigree* cujos pais e avós sejam conhecidos e *todos* tenham *pedigree*.

d) Defina uma função que calcula o número de gerações conhecida de um dado cão cujo número de registo é passado como argumento

```
int geracoes( AG , int);
```