Teste

Programação Imperativa
1º Ano – LEI/LCC

20 de Junho de 2012
Duração: 2 horas

Parte I

Esta parte do teste representa 12 valores da cotação total. Cada alínea está cotada em 2 valores. **A não obtenção de uma classificação mínima de 8 valores nesta parte implica a reprovação no teste.**

1. Considere as seguintes definições para representar uma pauta

```
#define Max 300
typedef struct linha {
    int numero;
    char nome [100];
    int nota; // de 0 a 20
} Pauta [Max];
```

   Defina uma função `int maisFreq (Pauta p, int n)` que calcula a nota mais frequente entre os `n` primeiros alunos da pauta `p`.

2. Defina uma função `int toBits (unsigned int x, char b[32])` que preenche o array `b` com a representação binária do número `x`. A função deverá retornar o número de bits 1 que foram escritos.

   Por exemplo, a invocação `toBits (54, v)` deverá colocar em `v` a string

   `"00000000000000000000000000101100"`\0

   e retornar 3 (note que a representação binária de 54 é 101100).

3. Defina uma função `int comuns (int a[], int na, int b[], int nb)` que calcula quantos elementos distintos os vectores `a` (com `na` elementos) e `b` (com `nb` elementos) têm em comum.

   Assuma que os vectores `a` e `b` estão ordenados por ordem crescente sem elementos repetidos.

4. A função `char *strdup (char s[])`, pré-definida em C, aloca memória suficiente para armazenar uma cópia da string `s`, faz essa cópia e retorna o endereço da cópia.

   Apresente uma possível definição da função `strdup`.

5. Considere a seguinte definição para representar listas ligadas de inteiros e de pares de inteiros

```
typedef struct par {
    int x, y;
    struct par *seg;
} Par, *ListaPar;

typedef struct um {
    int v;
    struct um *seg;
} Nodo, *ListaUn;
```

   (a) Defina uma função `int fechada (ListaPar l)` que testa se numa dada lista não vazia o primeiro elemento da lista é igual ao último.

   (b) Relembre a função `unzip` do Haskell que constroi duas listas a partir de uma lista de pares. Defina em C a função `int unzip (ListaPar l, ListaUn *x, ListUn *y)` que coloca nos endereços `x` e `y` o resultado de fazer o `unzip` de `l`. A função deverá retornar o comprimento das listas produzidas.

Parte II

1. Defina uma função `int iguaisConsecutivos (char s[])` que, dada uma string `s` calcula o comprimento da maior sub-string com caracteres iguais.

   Por exemplo, `iguaisConsecutivos ("aabcccaac")` deve dar como resultado 3, correspondendo à repetição `"ccc"`.

2. Defina uma função `ListaPar filtra (ListaPar l, int s)` que remove da lista de pares `l` todos os pares cuja soma é superior ou igual a `s`. Não se esqueça de libertar a memória correspondente a esses pares.

3. Considere o seguinte tipo para representar árvores binárias de inteiros.

```
typedef struct um {
    int v;
    struct um *esq, *dir;
} Nodo, *ArvUn;
```

   (a) Defina uma função `void freeArv (ArvUn a)` que liberta a memória ocupada por uma árvore binária.

   (b) Defina uma função `int nivel (ArvUn a, int n, int v[])` que preenche o vector `v` com os elementos de `a` que se encontram no nível `n`.

   Considere que a raíz da árvore se encontra no nível `1`.

   A função deverá retornar o número de posições preenchidas do array.