MAC122/BCC **PROVA SUBSTITUTIVA (SEMI-ABERTA)** *10/12/98*

Q1. [3 pontos] Suponha que os nós de uma árvore binária t são do tipo `No`, declarado assim:

```
typedef struct no_struct {
  char *chave;
  long peso;
  struct no_struct *esq, *dir;
} No;
```

Estamos interessados em determinar a *soma ponderada das profundidades das folhas* de t, definida como sendo

$$\mathrm{soma\_pond}(t) = \sum_x \mathrm{peso}(x)\mathrm{prof}(x),$$

onde a soma se estende sobre todas as **folhas** x de t (isto é, nós sem nenhum filho) e, naturalmente, $\mathrm{peso}(x)$ e $\mathrm{prof}(x)$ denotam, respectivamente, o valor do campo `peso` e a profundidade de x em t. Escreva uma função de protótipo

```
long soma_pond(No *t, long *peso_total);
```

que recebe em `t` um ponteiro para a raiz de uma tal árvore e devolve a soma ponderada das profundidades das folhas desta árvore. Use o parâmetro `peso_total` para devolver o peso total das folhas desta árvore.

Q2. [3 pontos] Nesta questão, estamos interessados no mergesort para listas ligadas *com caudas*. Considere o seguinte segmento de código:

```
No *b, *c, *aa=d, /* d aponta para o começo de uma lista */
   *bb=z, *cc=z; /* z aponta para a cauda das listas */
if (d->prox!=z) {
  aa=b=c=d;
  d=d->prox->prox->prox;
  while (d!=z) {
    b=b->prox;
    c=c->prox->prox;
    d=d->prox->prox->prox;
  }
  bb=b->prox; b->prox=z;
  cc=c->prox; c->prox=z;
```

Após a execução do código acima, temos três listas, cujos elementos iniciais são apontados por `aa`, `bb`, e `cc`.

(i) Determine o número de elementos que estas três listas vão ter quando a lista original (dada por `d`) tem, respectivamente, 0, 1, 2, 3, 4, 5, e 6 elementos. Em todas as contagens, *não* conte a cauda!

(ii) Idem para o caso geral, em que a lista original tem $n \geq 0$ elementos.

(iii) Escreva uma versão de mergesort para listas que usa o trecho de código acima (ou alguma variante próxima). A sua função deve fazer três chamadas recursivas dela mesma. Você deve escrever explicitamente a função `merge()` correspondente.

Q3. [2 pontos] Suponha que estamos utilizando uma árvore binária de busca (ABB) para manter um conjunto de palavras. Estamos interessados em fazer buscas neste conjunto de palavras.

(i) Qual é a relevância da altura desta árvore?

(ii) Qual é a relevância da média das profundidade dos nós desta árvore?

(iii) Suponha que inserimos, nesta ordem, as palavras `abobora`, `laranja`, `jabuticaba`, `melancia`, `papaya`, `acerola`, `graviola`, e `pera` em uma ABB originalmente vazia. Desenhe o estado desta árvore após cada inserção.

(iv) Determine os dois parâmetros mencionados em (i) e (ii) acima para esta árvore.

Q4. [2 pontos] Suponha que estamos utilizando uma tabela de hashing para manter um conjunto de palavras. Estamos interessados em fazer buscas neste conjunto de palavras.

(i) O que é o fenômeno da "colisão"? O que é "resolução de colisões por encadeamento"?

(ii) Como podemos utilizar a idéia de reorganização de listas para tirar vantagem de que certas palavras são buscadas mais freqüentemente que outras?