PROVA SUBSTITUTIVA DE ESTRUTURAS DE DADOS
BCC, 1o. SEMESTRE DE 2008

Instruções:

1. Não destaque as folhas do caderno de soluções.
2. A prova pode ser feita a lápis. Cuidado com a legibilidade.
3. Não é permitido o uso de folhas avulsas para rascunho.
4. Não é necessário apagar rascunhos no caderno de soluções.
5. Asserções imprecisas valem pouco. Justifique suas asserções (dentro do razoável!).

Faça no máximo 3 questões!

1. [3 pontos]
   (i) Suponha que inserimos, nesta ordem, as 11 chaves `Q U E M I T O F A C L` em uma árvore binária de busca (ABB) inicialmente vazia. Desenhe as ABBs que temos ao longo do processo (desenhe 11 árvores).
   (ii) Suponha agora que removemos `E` da ABB resultante em (i). Mostre como se dá esse processo de remoção.

```
link partR(link h, int k)
  { int t = h->l->N;
    if (t > k) { h->l = partR(h->l, k); h = rotR(h); }
    if (t < k) { h->r = partR(h->r, k-t-1); h = rotL(h); }
    return h; /* rotR() e rotL() ajustam o campo N dos nós */
  }
link joinLR(link a, link b)
  { if (b == z) return a;
    b = partR(b, 0); b->l = a;
    return b;
  }
link deleteR(link h, Key v)
  { link x; Key t = key(h->item);
    if (h == z) return z;
    if (less(v, t)) h->l = deleteR(h->l, v);
    if (less(t, v)) h->r = deleteR(h->r, v);
    if (eq(v, t)) { x = h; h = joinLR(h->l, h->r); free(x); }
    return h;
  }
void STdelete(Key v)
  { head = deleteR(head, v); }
```

2. [3 pontos] Suponha que inserimos, nesta ordem, as 11 chaves `Q U E M I T O F A C L` em uma ABB aleatória inicialmente vazia. Suponha que a árvore resultante é tal que todas as subáraves esquerdas são vazias (todos os ponteiros esquerdos apontam para `z`).
   (*i*) Mostre como se deu essa evolução da ABB vazia até esta ABB resultante (desenhe as várias ABBs intermediárias).
   (*ii*) Qual é a probabilidade de termos essa ABB resultante? Não se esqueça de justificar sua resposta com cuidado.

```
link insertT(link h, Item item)
  { Key v = key(item);
    if (h == z) return NEW(item, z, z, 1);
    if (less(v, key(h->item))) { h->l = insertT(h->l, item); h = rotR(h); }
                          else { h->r = insertT(h->r, item); h = rotL(h); }
    /* Lembre que rotR() e rotL() ajustam o campo N dos nós */
    return h;
  }
link insertR(link h, Item item)
  { Key v = key(item), t = key(h->item);
    if (h == z) return NEW(item, z, z, 1);
    if (rand()< RAND_MAX/(h->N+1)) return insertT(h, item);
    if less(v, t) h->l = insertR(h->l, item);
             else h->r = insertR(h->r, item);
    (h->N)++; return h;
  }
void STinsert(Item item)
  { head = insertR(head, item); }
```

3. [4 pontos]
   (*i*) Suponha que inserimos, nesta ordem, as chaves `A S E R C H I N G X` em uma árvore 2-3-4 inicialmente vazia. Desenhe as árvores 2-3-4 que temos ao longo do processo. Desenhe pelo menos 10 árvores; para dizer como a árvore evolui, voce pode (deve?) desenhar árvores intermediárias/auxiliares.
   (*ii*) Suponha agora que inserimos, nesta ordem, as chaves `A S E R C H I N G X` em uma árvore rubro-negra esquerdista (ARNE) inicialmente vazia. Desenhe as ARNEs correspondentes às últimas 4 árvores 2-3-4 do item (*i*) (desenhe as 4 árvores resultantes da inserção de `I`, `N`, `G` e `X`).
   (*iii*) Descreva cuidadosamente como ocorre a inserção de `G` na sua ARNE (queremos saber aqui como a 2a. ARNE que você desenhou no item (*ii*) acima evolui para se tornar a 3a. ARNE que você desenhou). Para responder a este item, desenhe várias ARNEs (ou partes de ARNE) para ilustrar a execução do algoritmo de inserção em ARNEs.
   (*iv*) Repita (*iii*) acima com a inserção de `X`: diga como a 3a. ARNE que você desenhou em (*ii*) evolui para a 4a. ARNE que você desenhou.

```
link RBinsert(link h, Item item)
{  Key v = key(item);
   if (h == z)
      return NEW(item, z, z, 1); /* o 1 indica ponteiro vermelho */
   if (h->l->red && h->r->red)) colorFlip(h);
   if (less(v, key(h->item))) h->l = insert(h->l, item);
                         else h->r = insert(h->r, item);
   if (h->r->red) h = rotL(h);
   if (h->l->red && h->l->l->red) h = rotR(h);
   return h;
}
void STinsert(Item item)
{  head = RBinsert(head, item); head->red = 0; }
```

4. [3 pontos] Suponha que inserimos, nesta ordem, as 11 chaves `Q U E M I T O F A C L` em uma tabela de hashing inicialmente vazia com $M = 12$ entradas. Suponha que usamos a função de hashing $13k \bmod M$ para transformar a k-ésima letra do alfabeto no endereço da tabela (por exemplo, `A` é levado ao endereço 1 e `S` é levado ao endereço 7).
   (i) Calcule o valor da função de hashing para cada uma das 11 chaves acima.
   (ii) Mostre claramente a evolução da tabela nesse processo, supondo que usamos resolução de colisões por *linear probing*.
   (iii) Suponha agora que removemos `Q` da tabela resultante em (i). Descreva como ocorre essa remoção.

```
void STdelete(Item item)
  { int j, i = hash(key(item), M); Item v;
    while (!null(i))
      if eq(key(item), key(st[i])) break;
      else i = (i+1) % M;
    if (null(i)) return;
    st[i] = NULLitem; N--;
    for (j = i+1; !null(j); j = (j+1) % M, N--)
      { v = st[j]; st[j] = NULLitem; STinsert(v); }
  }
```