PROVA SUBSTITUTIVA (FECHADA) DE ESTRUTURAS DE DADOS
BCC, 1o. SEMESTRE DE 2013

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!).
6. **Duração da prova: 1 hora e 40 minutos**

1. [5 pontos] Esta questão trata de árvores rubro-negra esquerdistas (ARNEs).
   (*i*) Suponha que inserimos, nesta ordem, as chaves `22 77 11 88 33 66 55 99 44` em uma ARNE inicialmente vazia. Desenhe as 9 ARNEs que resultam das 9 inserções acima.
   (*ii*) Suponha que inserimos, nesta ordem, as chaves

   `111 122 133 144 155 166 177 188 199 200 211 222 233 244 255`

   em uma ARNE inicialmente vazia. Desenhe a ARNE final desse processo.
   (*iii*) Repita (*ii*) com a seqüência

   `255 244 233 222 211 200 199 188 177 166 155 144 133 122 111.`

   (*iv*) Suponha que implementamos uma tabela de símbolos usando ARNEs. Escreva uma função em C de protótipo `Item STfloor(Key x)` que devolve o item `item` *de maior chave* na ARNE com `key(item)` $\leq$ `x` (caso não exista tal item, sua função deve devolver `NULLitem`). Para tanto, escreva e use uma função recursiva com protótipo `Item floorR(link h, Key x)`, que resolve este problema para a subárvore enraizada em `h`.

   Nos itens (*i*)–(*iii*), *não* desenhe árvores 2–3, desenhe as ARNEs diretamente. As duas rotinas principais envolvidas na inserção em ARNEs são dadas a seguir.

```
link LLRBinsert(link h, Item item)
  { Key v = key(item);

    /* Insert a new node at the bottom*/
    if (h == z) return NEW(item, z, z, 1);

    if (less(v, key(h->item))) hl = LLRBinsert(hl, item);
                          else hr = LLRBinsert(hr, item);

    /* Enforce left-leaning condition */
    if (hr->red && !hl->red) h = rotL(h);
```

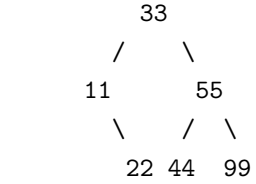
Date: Versão de 25 de junho de 2013.

```
    if (hl->red && hll->red) h = rotR(h);
    if (hl->red && hr->red) colorFlip(h);

    return h;
  }

void STinsert(Item item)
  { head = LLRBinsert(head, item); head->red = 0; }
```

2. [5 pontos] Suponha que inserimos, nesta ordem, as 6 chaves `11 22 33 44 55 99` em uma ABB aleatória inicialmente vazia. Suponha que a árvore resultante foi a árvore a seguir:



   (*i*) Mostre como se deu essa evolução da ABB vazia até esta ABB resultante (desenhe as várias ABBs intermediárias).

   (*ii*) Determine a probabilidade de termos essa ABB resultante com base no algoritmo de inserção em ABBs aleatorizadas (as rotinas são dadas a seguir). Mostre como você chegou a seu resultado.

```
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); }
    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); }
```

(*iii*) Suponha agora que estamos usando uma ABB *padrão* (*não-aleatorizada*) e que inserimos as 6 chaves `11 22 33 44 55 99` em uma tal ABB inicialmente vazia em alguma ordem. Podemos fazer isso de 6! jeitos diferentes, pois há 6! ordenações daquelas chaves. Descreva quais dessas 6! ordenações resultam na árvore dada no diagrama acima. Explique sua resposta.

(*iv*) Explique a relação entre suas respostas para (*ii*) e (*iii*) acima.