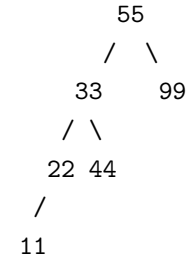
**PROVA 2 DE ESTRUTURAS DE DADOS**
**BCC, 1o. SEMESTRE DE 2012**

**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).

1. [4 pontos] Suponha que inserimos, nesta ordem, as 6 chaves `33 11 44 55 99 22` 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;
  }
```

*Date*: Versão de 11 de julho de 2012.

```
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.

2. [4 pontos] Esta questão trata de árvores rubro-negra esquerdistas (ARNEs).

(*i*) Suponha que inserimos, nesta ordem, as chaves `22 11 66 33 55 88 44 77 99` em uma ARNE inicialmente vazia. Desenhe as 9 ARNEs que resultam das 9 inserções acima.

(*ii*) Suponha que inserimos, nesta ordem, as chaves `11 22 33 44 55 66 77 88 99` em uma ARNE inicialmente vazia. Desenhe a ARNE final desse processo.

(*iii*) Repita (*ii*) com a seqüência `99 88 77 66 55 44 33 22 11`.

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

    return fixNr(h);
  }

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

3. [3 pontos] Queremos um programa `ind` que faz o seguinte: a entrada de `ind` será um arquivo texto, contendo um livro, com as páginas separadas por um caracter especial, por exemplo, `^L` (isto é, todo fim de página é marcado por uma linha que contém `^L` no início e nada mais). Ao receber essa entrada, `ind` deve gerar como saída um "índice remissivo completo", isto é, a lista de palavras distintas que ocorrem no livro dado, em ordem alfabética, com cada

palavra seguida de uma lista de números, indicando em quais páginas essa palavra ocorre: se a palavra $w$ ocorre nas páginas de número $p_1, p_2, \ldots$, então a linha

$w \quad p_1, p_2, \ldots$

deve ocorrer nesse índice.

($i$) Descreva, muito brevemente e em linhas gerais, como você poderia implementar `ind`. Se um livro $T$ tem $N$ palavras no total e $M$ palavras distintas, quanto tempo sua implementação de `ind` levará para processar $T$? (Responda usando a notação $O$.) Justifique sua resposta (naturalmente, você deve descrever uma implementação eficiente para `ind`).

($ii$) Suponha agora que `ind` recebe também como entrada uma lista ordenada $D$ de palavras que contém todas as palavras do português. Além de produzir o índice remissivo para o livro $T$, como descrito acima, `ind` deve também produzir a lista de palavras que *ocorrem* em $T$ mas *não ocorrem* em $D$ (assim, `ind` serve como um verificador ortográfico). Descreva, em algum detalhe, como você implementaria essa parte adicional em `ind`. Quanto tempo sua implementação de `ind` levaria para executar *essa tarefa adicional*, supondo que $T$ tem $N$ palavras no total, $M$ palavras distintas, e $D$ tem $P$ palavras? (Responda usando a notação $O$.) Justifique sua resposta (naturalmente, você deve descrever uma implementação eficiente para essa parte adicional também).

($iii$) Suponha que `ind` seja implementado como você descreveu em ($i$) e ($ii$) acima. Com base na experiência que você adquiriu com os EPs dessa disciplina, estime quanto tempo ele levará para processar um livro com, digamos, $N = 300.000$ e $M = 30.000$ (esses são, aproximadamente, os parâmetros para *Ulysses*, de James Joyce), supondo que $D$ tenha 3.000.000 palavras (esse é um número arbitrário). Sua estimativa pode ser algo como "poucos segundos", "poucos minutos", "alguns minutos", etc. Relate experiências concretas que você teve que o levam a dar esta estimativa.