PROVA 2 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 `33 11 44 22 55 99 66 88 77` 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`.
   (*iv*) Suponha que implementamos uma tabela de símbolos usando ARNEs. Escreva uma função em C de protótipo `Item STceil(Key x)` que devolve o item `item` *de menor chave* na ARNE com `x` $\leq$ `key(item)` (caso não exista tal item, sua função deve devolver `NULLitem`). Para tanto, escreva e use uma função recursiva com protótipo `Item ceilR(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);
    if (hl->red && hll->red) h = rotR(h);
    if (hl->red && hr->red) colorFlip(h);

    return h;
```

Date: Versão de 18 de junho de 2013.

```
}

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

2. [5 pontos] Esta questão trata de *hashing* com *probing* sequencial/linear. Como de usual, escrevemos M para o tamanho da tabela.

   (i) Suponha que nossas chaves são as letras `A,...,Z`, e que a k-ésima letra é mapeada para o endereço $16k\%M$ (por exemplo, `B` é mapeado para o endereço $16 \times 2\%M = 32\%M$, enquanto que `K` é mapeado para $16 \times 11\%M = 176\%M$). Suponha $M = 15$. Suponha que inserimos as chaves `E A S Y Q U T I O N` em uma tabela inicialmente vazia. Mostre como evolui a tabela (faça um diagrama para cada uma das 10 inserções).

   (ii) Refaça o item (i), supondo agora $M = 11$.

   (iii) Suponha agora que inserimos N objetos em uma tabela de *hashing* inicialmente vazia de tamanho M (usando *probing* sequencial; supomos $N < M$). Suponha que vale a hipótese (H): *os N endereços dados pela função de hashing são variáveis independentes, uniformemente distribuídas em* $0, \ldots, M-1$. Sabemos que, nesse caso, ao fazermos uma busca com sucesso, fazemos em média

   $$\sim \frac{1}{2}\left(1 + \frac{1}{1-\alpha}\right) \tag{1}$$

   probes (isto é, examinamos essa quantidade de endereços da tabela de *hashing*). No caso de busca sem sucesso, fazemos em média

   $$\sim \frac{1}{2}\left(1 + \frac{1}{(1-\alpha)^2}\right) \tag{2}$$

   probes. Nas expressões acima, α é o fator de carregamento da tabela: $\alpha = N/M$. Calcule o valor das expressões acima quando $\alpha = 1/2$, $\alpha = 2/3$ e $\alpha = 9/10$.

   (iv) Suponha que usamos a função de *hashing*

```
int hash(char *v, int M)
  { int h = 0, a = 256;
    for (; *v != '\0'; v++)
      h = (a*h + *v) % M;
    return h;
  }
```

   Suponha $M = 256$. Suponha que inserimos $N = 128$ itens em nossa tabela, e que, por uma impressionante coincidência, todos os itens têm chaves que terminam com as letras `b, d, f, h, j, l, n, p, r` (o valor ASCII de `b` é 98 (decimal)), com pelo menos dois itens com chaves que terminam em cada uma das letras acima (isto é, pelo menos dois itens com chaves que terminam em `b`, pelo menos dois itens com chaves que terminam em `d`, etc). Fazemos agora uma busca sem sucesso da chave `bob` nessa tabela com fator de carregamento $\alpha = 1/2$. O que ocorre? Por exemplo, quantos *probes* ocorrem?