Tipos e Estruturas de Dados
ICMC-USP

Aluno: _____Nro.USP_____

2ª. Prova – 22/05/07

1) Seja a ED de uma tabela hash dada ao lado. Foi adotado para inserção o processo de espalhamento linear, ou seja, havendo uma colisão em h (índice retornado pela função hash(Chave) fornecida), procura-se por uma nova posição de inserção nos índices consecutivos, ou seja, h+1, h+2, etc. Ciente disto, faça:

```
#define TamHash N      // N é primo
typedef int Chave;
typedef struct item {
   Chave ch;
   …..
}Dado;
typedef Dado TabHash[TamHash];
```

- Escreva a função int Insere(TabHash t, Dado d) que coloca o dado d na tabela hash t. Não se pode inserir chaves repetidas. *Overflow* deve ser notificado. **(1,5)**
- O espalhamento linear provoca, especialmente quando a tabela está com a metade ocupada, o aparecimento de clusters, ou seja, blocos consecutivos alocados em posições adjacentes. De que forma o espalhamento quadrático minimiza este problema ? Justifique **(0,5)**

2) Seja a estrutura do nó ao lado de uma árvore binária. Escreva:

```
typedef struct no {
   Dado dado;
   struct no *esq;
   struct no *dir;
}No;
```

- Uma função que calcule a altura de uma árvore binária. Por convenção, a altura de uma árvore vazia é -1. A altura de uma árvore com um único nó é zero. **(1,0)**
- Uma função NÃO recursiva que percorra uma árvore binária com chaves inteiras, em Inordem. Voce tem a sua disposicao uma pilha P de tamanho ilimitado com as funções: int pop(), push(P, valor) que mantêm o topo atualizado. P.topo==0 indica pilha vazia. **(1,0)**

3) Seja a seqüência: 1, 20, 3, 18, 5, 16, 7, 14.

- Construa uma árvore AVL. Toda vez que a inserção de um nó provocar a necessidade de uma rotação, ilustre a árvore anterior e posterior à operação, indicando qual foi (ou foram) as rotações necessárias em cada caso. **(1,0)**
- Da árvore resultante do passo acima, retire as chaves 5, 16 e 20 (nesta ordem) ilustrando, a exemplo do item acima, as rotações ocorridas. Obs. Considere, na remoção, a substituição pelo antecessor da chave. **(1,0)**

4) Construa a árvore-B de ordem 4 resultante da inserção da seguinte seqüência de chaves (na ordem dada), usando o algoritmo de inserção visto em aula. Obs: ordem 4 significa um máximo de 4 nós filhos.

a) 45 80 90 95 70 15 25 30 33 8 **(1,0)**
b) A partir da árvore obtida em (a), realize as operações de remoção dos nós abaixo, fornecendo as configurações intermediárias da árvore-B: 90 95 80 (nesta ordem) **(1,0)**

5) Escreva uma função que percorra uma árvore B, listando todos os nós por ordem de chave. Dica: pense no procedimento InOrdem de uma árvore binária. Utilize a estrutura de árvore-B como vista em sala, dada ao lado. cont contém o nro de chaves existentes em um nó. (2,0).

```
#define MAX m-1
#define MIN (m/2) - 1
typedef struct no {
   Dado d[MAX+1];
   struct no *g[MAX+1];
   int cont;
}No;
 m é a ordem da árvore
```

Boa prova a todos