PUC-Rio – Estruturas de Dados – INF1010
Prova 1 – Turma 3wa – 10/4/2019

Responda as questões abaixo nas folhas em separado distribuídas junto com a prova. As respostas podem ser escritas a lápis ou caneta. Indique claramente a questão que está respondendo e, quando cabível, os passos que seguiu para chegar a sua resposta.

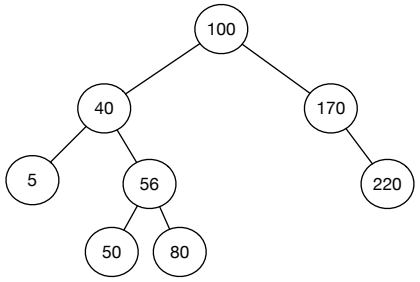
Por favor guarde seu celular/tablet/o_que_for antes de começar a prova e não o utilize até sair da sala.

1. (1 ponto) Considere a interface para operações sobre conjuntos de números inteiros vista em aula, esboçada abaixo:

```
typedef unsigned int Set;
/* cria um conjunto com n elementos */
Set* setCreate(void);
/* insere o elemento i no conjunto */
void setInsert(Set *set, int i);
/* remove o elemento i do conjunto */
void setRemove(Set *set, int i);
...
```

   Vamos supor uma implementação que contempla conjuntos de números naturais com valores entre 0 e 31, e que utiliza a representação por mapa de bits vista em sala. Implemente a função `setRemove`.

2. (1 ponto) Suponha a árvore AVL mostrada a seguir.



   Para cada uma das operações a seguir, aplicada sobre a árvore original, mostre a árvore resultante.

   (a) inserção de 250

   (b) inserção de 45

3. Considere a representação a seguir para árvores binárias de busca.

```
typedef struct smapa Mapa;
struct smapa {
  int chave;
  int dados;
  Mapa* esq;
  Mapa* dir;
};
```

   (a) (1 ponto) Escreva uma função C que, dado um mapa m com essa representação e valores inteiros min e max, mostre, em ordem crescente, todas as chaves presentes no mapa com valores entre min e max (exclusive).

```
void mostrachavesint  (Mapa *m, int min, int max);
```

(b) (1 ponto) Sua função evita chamadas recursivas desnecessárias? Caso não, modifique-a para que o faça. (Caso já tenha construído a função assim, não precisa fazer mais nada.)

4. (2 pontos) Supondo a mesma representação de árvore binária usada na questão anterior, explique o que faz o código a seguir:

```
int foo (Mapa *m) {
  if (m==NULL) return 0;
  else if (m->esq==NULL && m->dir==NULL) return 1;
  else return foo(m->esq) + foo(m->dir);
}
```

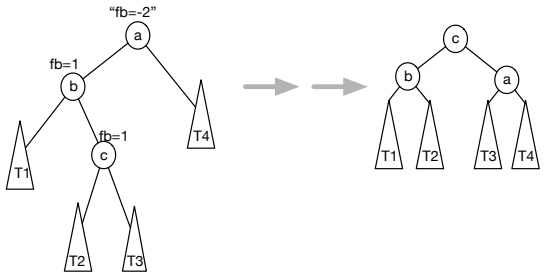
5. Suponha uma implementação de AVL com a representação a seguir.

```
typedef struct smapa Mapa;
struct smapa {
  int chave;
  int conteudo;
  short int bf;
  struct smapa* esq;
  struct smapa* dir;
};
...
static Mapa* rotacao_a_direita (Mapa *m);
```

(a) (1 ponto) Escreva em C a operação `rotacao_a_direita` convencional.

(b) (1 ponto) Suponha agora que acrescentamos um campo `int numnos` à struct `smapa`, contendo o número de nós na árvore abaixo desse nó. Modifique o código da operação `rotacao_a_direita` para que a operação corrija os valores desse campo apropriadamente.

6. (2 pontos) Considere a operação de rotação dupla mostrada abaixo, para correção de árvore AVL desbalanceada. Diga quais serão os fatores de balanceamento finais dos nós a, b e c, *mostrando* como podem ser calculados a partir da configuração inicial.



Boa prova!