PUC-Rio – Estruturas de Dados – INF1010
Prova 1 – Turma 3wb – 13/09/2017

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.

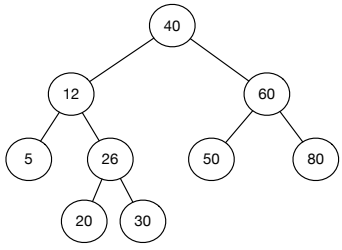
1. (1 ponto) Considere uma implementação de conjuntos de números naturais que contempla conjuntos de números com valores entre 0 e 31, e que utiliza a representação por mapa de bits vista em sala. Implemente a função `setRemove` *sem utilizar* outras operações da interface apresentada.

```
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);
...
```

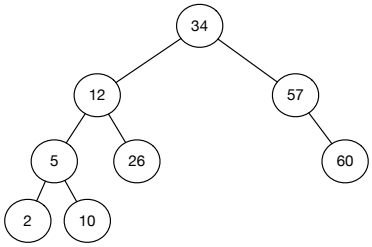
2. (2 pontos) Suponha que queremos escrever uma função que retorne o dado associado a uma chave em um mapa implementado por uma árvore binária. Considerando a representação de árvores binárias usada nos laboratórios, analise a função busca mostrada a seguir.

```
int busca(Mapa* m, int c) {
  while (m != NULL && m->chave > c)  m = m->esq;
  while (m != NULL && m->chave < c)  m = m->dir;
  if (m==NULL) return -1;
  else return m->dados;
}
```

Explique por que a função não é correta, dando um exemplo de chave que não seria encontrada na árvore binária a seguir, apesar de estar nela. Diga o que a função retornaria para essa chave, sabendo que o valor do dado é sempre igual ao da chave.



3. (2 pontos) Suponha a árvore AVL mostrada a seguir.



Como ficará a árvore se realizarmos cada uma das operações (*atenção:* cada uma das operações indicadas ocorre na árvore original mostrada!). Desenhe (na folha a parte) a nova árvore para cada caso.

(a) inserção com chave 65

(b) inserção com chave 1

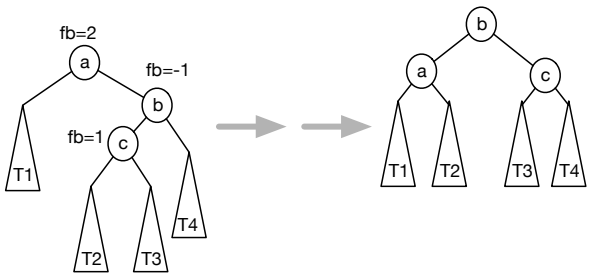
(c) inserção com chave 11

(d) remoção da chave 57

4. (2 pontos) Considere a função `e_abb`, vista em laboratórios, que determina se uma determinada árvore binária é uma árvore binária de busca, isto é, se para cada nó n da árvore, todas as chaves da subárvore à sua esquerda são menores que a chave de n, e todas as chaves na sua subárvore à direita são maiores que a chave de n. Uma outra forma de implementar `e_abb`, diferente da discutida em sala, é escrever uma função que recebe dois parâmetros, `min` e `max`, que indicam os valores mínimo e máximo permitidos na subárvore a ser visitada a cada chamada recursiva. Escreva a função `e_abbrecursiva` que implementa essa idéia. Considere a representação de árvore binária usada nos laboratórios.

```
static int e_abbrecursiva (Mapa* m, int min, int max) {
  ...
}
int e_abb(Mapa *m) {
  return(e_abbrecursiva(m, INT_MIN, INT_MAX));
  /* INT_MIN e INT_MAX são, respectivamente, o menor e maior inteiro
     representáveis na máquina utilizada.  */
}
```

5. (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, *explicando* como podem ser calculados a partir da configuração inicial.



6. (2 pontos) Escreva uma função `mostracaminholongo` que imprima todas as chaves em nós no caminho mais longo da raiz até uma folha de uma árvore AVL. Se existirem vários caminhos de mesmo tamanho, a função pode imprimir qualquer um deles. Suponha a representação de árvore usada nos laboratórios. Sua função pode ser recursiva ou não.

```
void mostracaminholongo (Mapa *m);
```

Boa prova!